# A DYNAMIC SYSTEMS TAKE HOME LABORATORY

By

TREVOR ECKERT

Bachelor of Science in Electrical Engineering

Oklahoma State University

Stillwater, Oklahoma, United States of America
2016

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2018

A DYNAMIC SYSTEMS TAKE HOME LABORATORY

Thesis Approved:

---

Dr. Martin Hagan

Emeritus Professor

Department of Electrical and Computer Engineering

Thesis Advisor

---

Dr. Keith Teague

Professor

Department of Electrical and Computer Engineering

Committee Chair

---

Dr. Carl Latino

Associate Professor

Department of Electrical and Computer Engineering

ACKNOWLEDGMENTS

I would like to first thank my thesis advisor, Dr. Martin Hagan. His knowledge, patience, and guidance made it possible for me to conduct my research and write this thesis. Without him, I would not have grown and learned as much as I have. I would also like to thank Dr. Amir Jafari, one of Dr. Martin Hagan′s PhD students. His assistance with the Take Home Labs helped immensely.

Also, I would like to thank my parents, Robert and Jill Eckert, and my siblings for their love and support during my master′s studies and research. I would also like to thank all of my friends for their support in writing this thesis.

Lastly, I would like to thank my grandpa, Paul Eckert, who always pushed his grandchildren to pursue their passions and education. Thank you for everything you have given to me. May he forever rest in peace. [1]

[1] Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: Trevor Eckert

Date of Degree: May, 2018

Title of Study: A DYNAMIC SYSTEMS TAKE HOME LABORATORY

Major Field: Electrical Engineering

Abstract: Laboratory experiments are important for engineering students to acquire hands-on experience and reinforce theoretical concepts. Current issues with lab experiments are that they take up space and can be expensive. Take Home Labs is a series of inexpensive experiments that students can perform at home. The increasing popularity of microcontrollers and 3-D printing makes it possible for the cost of the Take Home Labs to be less than the cost of a textbook. The software used for Take Home Labs is MATLAB/Simulink, which is available to most students at universities. These experiments were first introduced in the Dynamic Systems course in the fall of 2015. This thesis discusses the changes made to the experiments and why, based on the results from 2015. The work in this thesis also focuses on the assessment of the experiments in Dynamic Systems course in the fall of 2017. A web based survey question and answer platform called Piazza was used for the assessment. A new lab, Optimal State Feedback Control of a Rotary Flexible Link, was also created in this thesis.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

# CHAPTER 1

# INTRODUCTION

In engineering courses, experience with laboratory experiments are crucial for enhancing students′ ability to apply theoretical concepts to physical systems. Depending on the course, these lab experiments can be difficult to implement due to limited lab space, TA support, limited funding, and time for these labs. While most universities have lab spaces and time for circuits, digital logic design, and embedded systems programming, controls course labs can be lacking. Controls courses usually depend on homework and simulations to reinforce theoretical concepts. Due to the increased popularity of cheap microcontrollers, cheap sensors and other hardware, and 3D printing, there are possibilities to create take home laboratory experiments. This thesis discusses the development of inexpensive lab experiments that can be done at home.

The Take Home Labs (THL) concept was initially developed by Carion Pelton and Sean Hendrix [1] [2]. This thesis will discuss improvements that we made to THL, as well as extensive assessment of THL that was performed during the fall of 2017.

THL takes advantage of cheap microcontrollers, cheap DC motors, cheap sensors, and 3D printing. There is a website, http://thl.okstate.edu/, with all of the information on where to get certain hardware, 3D printing files, and detailed experiment handouts on how to put together and perform the labs (see Figure 1.1). There is already a list of experiments that refer to the following theoretical concepts: Sampling and Data Acquisition, Open Loop Step Response, Open Loop Frequency Response, Closed Loop Step

Response, Closed Loop Frequency Response, Root Locus, Pole Placement, and Optimal Control.

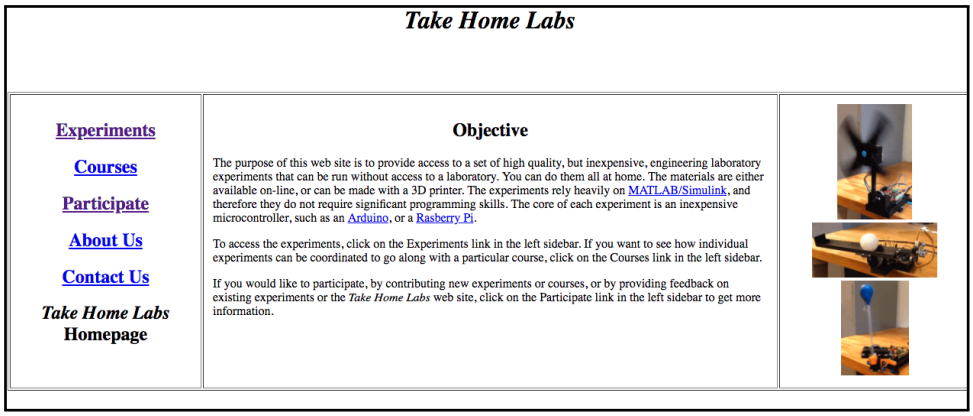



Figure 1.1: Take Home Labs Webpage

## 1.1 Literature Review

There have been other recent efforts in the development of laboratory experiments that can be done at home. In [3], a fourth-order, linear mass-spring-damper system and an analog filtering system were designed and packaged with a controller board for less than $100, and provided for students as at-home system kits. A take home hardware kit for a Digital Design Lab was developed for students in [4]. Two experiments that involve a DC motor/tachometer system and a heater/temperature sensor system were provided in [5]. A custom lab kit was developed that included a custom data acquisition board with a PIC microcontroller. The low cost hardware experiments were designed and used in several mechanical engineering courses, and were found to be effective based on student feedback. A low cost lab kit platform based on an Arduino microcontroller was designed for teaching technology of modern display systems [6]. Student surveys showed high praise for the overall lab kit, and specifically the Arduino platform. A low-cost rapid control prototyping system using MATLAB/Simulink and an Arduino is discussed in [7]. The Arduino and MATLAB are also used in [8] as part of the take home lab kit designed

by six mechanical engineering senior design students at the University of Minnesota. An electronics take home lab kit is described in [9, 10]. It consists of a custom circuit board and is designed to be compatible with LabVIEW.

A commercial approach to a real-time control design and implementation has been developed by Quanser [11]. Every Quanser system is compatible with MATLAB/Simulink and LabView. There are a wide variety of Quanser systems that are of high quality. Because their setups are of such high quality, they are also of high cost.

Minseg is a cheaper alternative that involves a self balancing robot [12]. This lab setup includes the use of an Arduino or Raspberry Pi. Simulink support libraries have been created by the developers along with experiments. Although cheap, this only offers one type of system (a mini segway) and this is limited on theoretical concepts.

In [13], control systems labs using a microcomputer is described. Important theoretical concepts are applied to physical electro-mechanical systems:

- Open Loop Step Response
- Closed Loop Step Response
- Open Loop Frequency Response
- Closed Loop Frequency Response
- Stability Analysis
- Root Locus Design

The hardware of the lab station consists of a DC motor, digital to analog (D/A) converter to interface the motor with the computer, a power amplifier for the output current, and a

sensor for the motor position. Since the students do not set up the lab stations, students have more time to attempt to understand and explain results obtained from performing the lab.

A modular controls lab was proposed in [14]. Students often fail to relate theoretical concepts such as root locus diagrams, Bode plots, and Nyquist diagrams to physical systems. Multiple systems were designed and used to apply these theoretical concepts. Six experiments were designed to provide a variety of dynamics:

- Ball on Beam
- Magnetically Levitated Sphere
- Car and Pole
- Wedge Balancer
- Two-Link Pendulum
- Inverted "T"

Students also were able to learn about real-time programming, effects of sampling rate, use of interrupts, importance of maintaining correct units, effects of nonlinearities (friction), and the iterative design process. The experiments were made modular to be able to maintain the lab stations as new technology changes.

In [15], remote labs used for controls is described. Students connect remotely to real-life plants that are kept in labs that are maintained by the professor and TAs. [15] describes a scheduling system where only one student can fill up a slot at a time to remote into the lab setup. This required an administrator, like a TA, to handle the functionality of the lab. The paper described two setups: the Stuart platform and a quadruple tank system. In [15], they also assessed the effectiveness of the labs by assessing the feedback

given by the students via Likert scale survey questions that covered comfort of the lab, usability, and the system as a whole. Upon the three-year assessment, their results appeared to be positive due to the increase of students, increase of test scores, and overall student interest in the remote labs.

## 1.2 THL Approach

THL was designed and developed to be affordable, adaptive, and easily accessible, providing real world applications for students to reinforce theoretical concepts. THL is much more expansive than other take home lab efforts. The vision is that it will be an open resource for a wide variety of STEM disciplines, although in this thesis we focus on control system courses.

The Take Home Labs are designed to be a series of experiments all using inexpensive open-source microcontrollers to make the labs very adaptable. MATLAB/Simulink is used to program the Arduino, so very little programming knowledge is required. MATLAB/Simulink is also very easy to obtain through universities. To make the labs even more affordable, 3D printing will be used for parts, because 3D printers are becoming more readily accessible to students. The website is simple and displays all information needed for completing the lab and gives the option for anyone to contribute for new experiments. These labs are small and therefore do not require any lab space. Instead, the experiments can be performed at home allowing the students to complete the labs on their own time. With all of these attributes, THL creates a cost effective, simple and readily accessible at-home lab experience.

### 1.3 2015 Experience

The Take Home Labs was first used in the ECEN 3723 class (Dynamic Systems) in the Fall of 2015. Students performed THL as their project, which counted for 25% of their grade. Seven labs were performed in total which included: Blinking LED, Simple DC Motor, Sampling and Data Acquisition, Open Loop Step Response, Closed Loop Step Response, Open Loop Frequency Response, and Root Locus Control Design. The Students′ lab notebooks are supposed to be a detailed report of all labs including theoretical calculations, discussions, and figures. Every two weeks, students turned in their lab notebooks, which included the lab that was to be completed within that two week period. This gave students a chance to make corrections based on the professor critiques, since a final grade on the project was not given until the final submission.

Since 2015 was the first time THL was used, it was expected to have some issues. In fact, there were three main issues: 1) the method for plotting measure data, 2) the length of the labs, and 3) the failure of students to complete all steps in the labs (especially the discussion sections).

Used throughout the labs, students used a library function provided by THL called serial plot to plot data collected during the experiments. The serial plot would read the serial port one byte at a time and combine adjacent bytes to produce the reading of the encoder on the DC motor. Students struggled and complained about getting the serial plot to work properly because of the "byte adjust" button. The "byte adjust" button would shift bytes and recalculate position to create meaningful data if the data did not appear to make sense. Once serial plot was introduced, we noticed that students seemed to have a negative attitude towards THL. Because of this we modified the software by removing the "byte adjust" button and instead the software adjusted the byte alignment automatically if the data was not valid.

Another major student complaint in 2015 was that THL took too much time. Students said they wanted another hour of credit for the course. We do not want THL to be a burden on the students, but instead a learning tool to reinforce theoretical concepts and get a basic understanding of physical limitations. The first change we made for 2017 was to reduce the number of labs from seven to five. We removed Blinking LED and Open Loop Frequency Response. We also removed some sections in some of the labs that distracted students from the main ideas. Also, in some of the labs there were a series of steps where students would repeat the same process with different values. A table was introduced instead. Now the students would complete the table while only having the process listed once in a smaller series of steps.

The last major issue was students skipping major steps in their lab reports. This may include theoretical calculations, hand sketches, and (especially) discussions. These steps are often the most important part of the labs. We want students to convey their work properly and be able to make comparisons among theory, simulation, and experiment. Excluding any aspect of the THL experiments makes it impossible to do this properly, since this is the main purpose of THL. To help with this issue, we decided to put more emphasis in the lab handouts on the importance of comparing theory, simulation, and experiment. We also put a summary table at the end of every lab handout with a list of every question or request for discussion that appeared throughout the lab. Students were reminded to check the list to be sure that they had answered all questions.

The issues described in this chapter were not the only issues, but the ones that seemed to have the most impact on the attitude of the students. It is impossible to tell exactly what changes had the largest impact, but this chapter is just a small summary of the overall experience in 2015. More detailed issues and changes will be discussed in future

chapters.

### 1.4 Thesis Outline

The remainder of the thesis will proceed as follows. Chapter 2 discusses a Social Question and Answer (SQA) web-based platform called Piazza and how we will use it to asses Take Home Labs. Chapters 3-7 are chapters that discuss the overall premise of the individual labs, changes made to the original labs and why, and the assessment of each experiment. Chapter 8 discusses the results from the assessment of all five labs. Chapter 9 describes a new lab developed called Optimal State Feedback Control (Rotary Flexible Link). Chapter 10 concludes with an overview of THL and the overall impact the labs had.

# CHAPTER 2

# PIAZZA AND ASSESSMENT

This chapter will go over Piazza, a web-based Social Question Answering (SQA) platform used in the 2017 Take Home Labs curriculum, and how we used Piazza to assess the Take Home Labs.

## 2.1 Piazza

Piazza is used in THL to give the students the chance to collaborate with fellow students, the instructor, and the TA. The goal behind Piazza is to provide students with the in-lab feel that the Take Home Labs do not otherwise give. Piazza is FERPA compliant and has the option for anonymous posting, which encourages students to be involved. Piazza has a good support team that replies quickly to the instructor questions which improves class fluidity.

Although Piazza is a very recent platform, there have been some studies on its effectiveness, including [16], which states "by using a SQA in an IS course, the students acquire skills required for IS profession such as collaboration, communication, analytical, and critical thinking." This was also investigated in [17], which concluded: "Findings suggest that micro-collaborations in SQA services promote all the three dimensions including knowledge, cognitive process, and social dimensions. The findings reveal that social dimension in micro-collaborations promote collaborative learning by enhancing community building, developing self-identity, and improving relational dynamics, which in turn support learning in various knowledge levels and improve the cognitive

process in learning."

Piazza is organized into four main tabs: Question and Answer, Resources, Statistics, and Manage Class. Figure 2.1 shows the tabs highlighted in red on the home page.

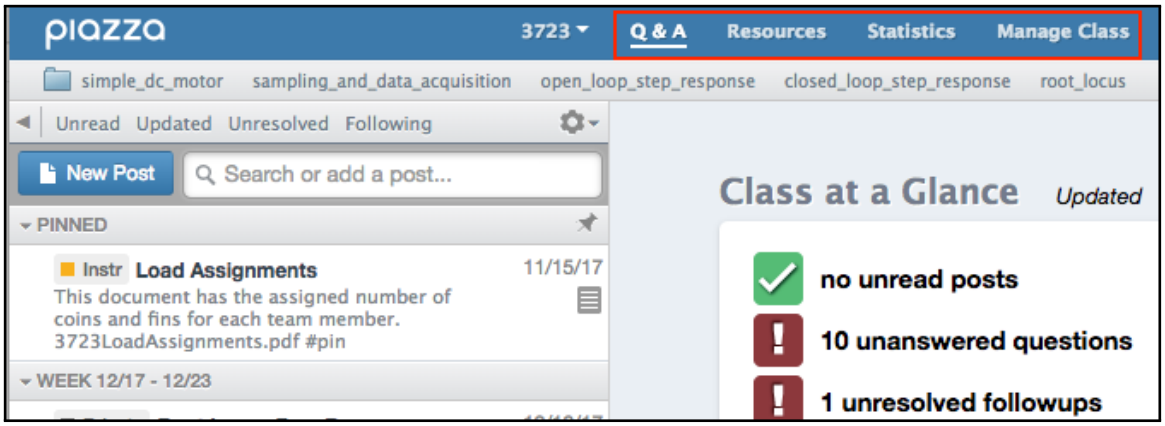



Figure 2.1: Tabs Highlighted In Red

### 2.1.1 Question and Answer

The Question and Answer tab is where the main functionality of Piazza is. Piazza can be organized into folders that separate different areas of a class. In our case, the folders were organized into each individual lab (see Figure 2.2). Individual posts, questions, and/or discussion can be created under each individual folder, making it easy to look back on responses for specific labs for students and results for instructors.

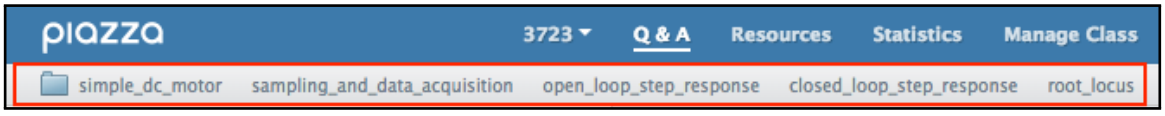



Figure 2.2: Folders Highlighted In Red

Posts to the Piazza page can be made by students, instructors, and TAs. When a new post is being made you can select who to post to, whether you want to post to the entire class or post to individual students/instructors. Then you select what folder the post

goes in. In every post there is a summary, which gets seen as the headline to the post. This is followed by the details, which, depending on the type of post, could be either a question or the start of a discussion. There are three different kinds of posts that can be made: question, note, and poll/in-class response. A question requires an answer. A note does not require an answer. Instead, it could be used for a type of announcement or the start of a discussion. A poll/in-class response is by far the most in-depth post that can be made. It has more options, and the end results show statistics. The poll/in-class response will be used for the pre-lab and post-lab questions, which will be discussed in the Assessment section. Some of the options for a poll/in-class response include: poll answer, poll type (one choice or multiple choice), when the poll closes, revotes, anonymity, how the results show, and when to make it visible to others. Figure 2.3 shows what drafting up a new note looks like. A note and a question look exactly the same on the drafting screen. Figures 2.4 and 2.5 show what drafting up a new poll/in-class response looks like.

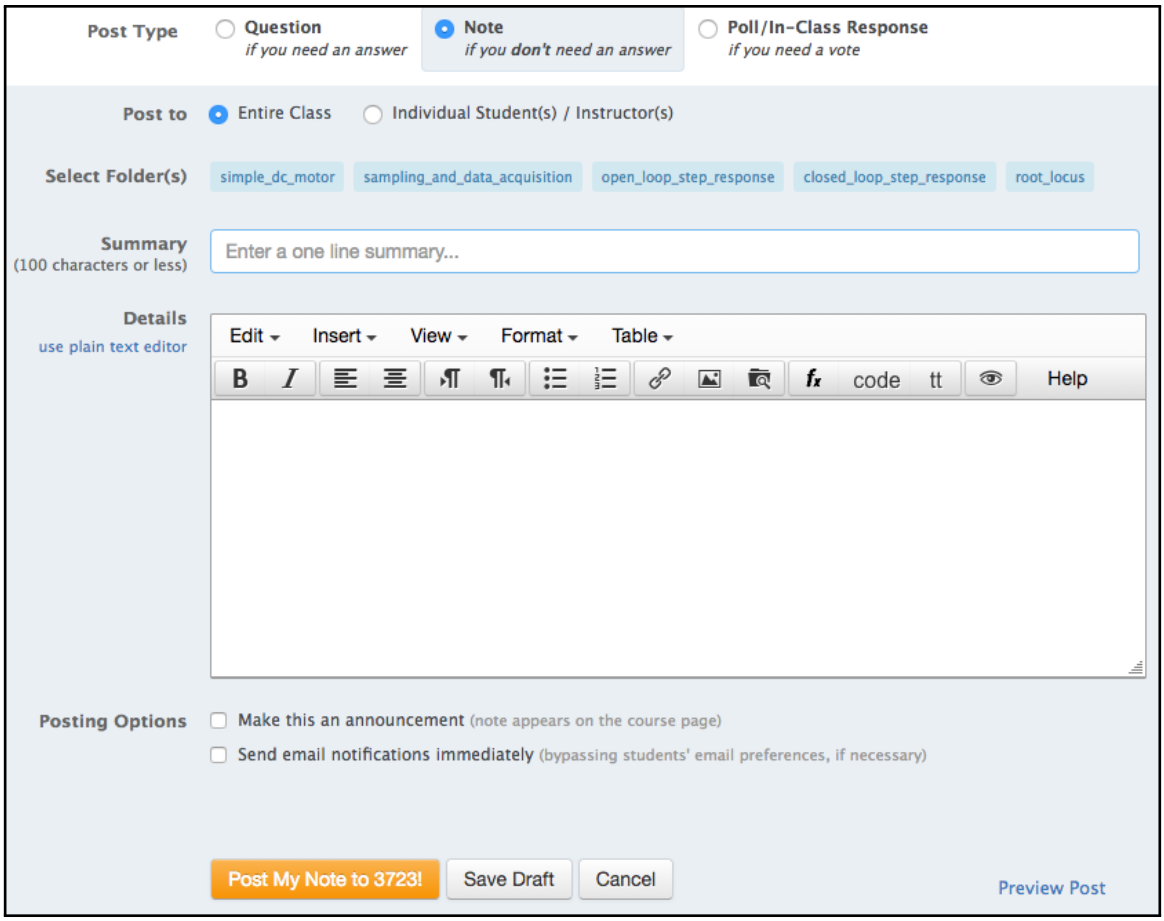



Figure 2.3: A New Note

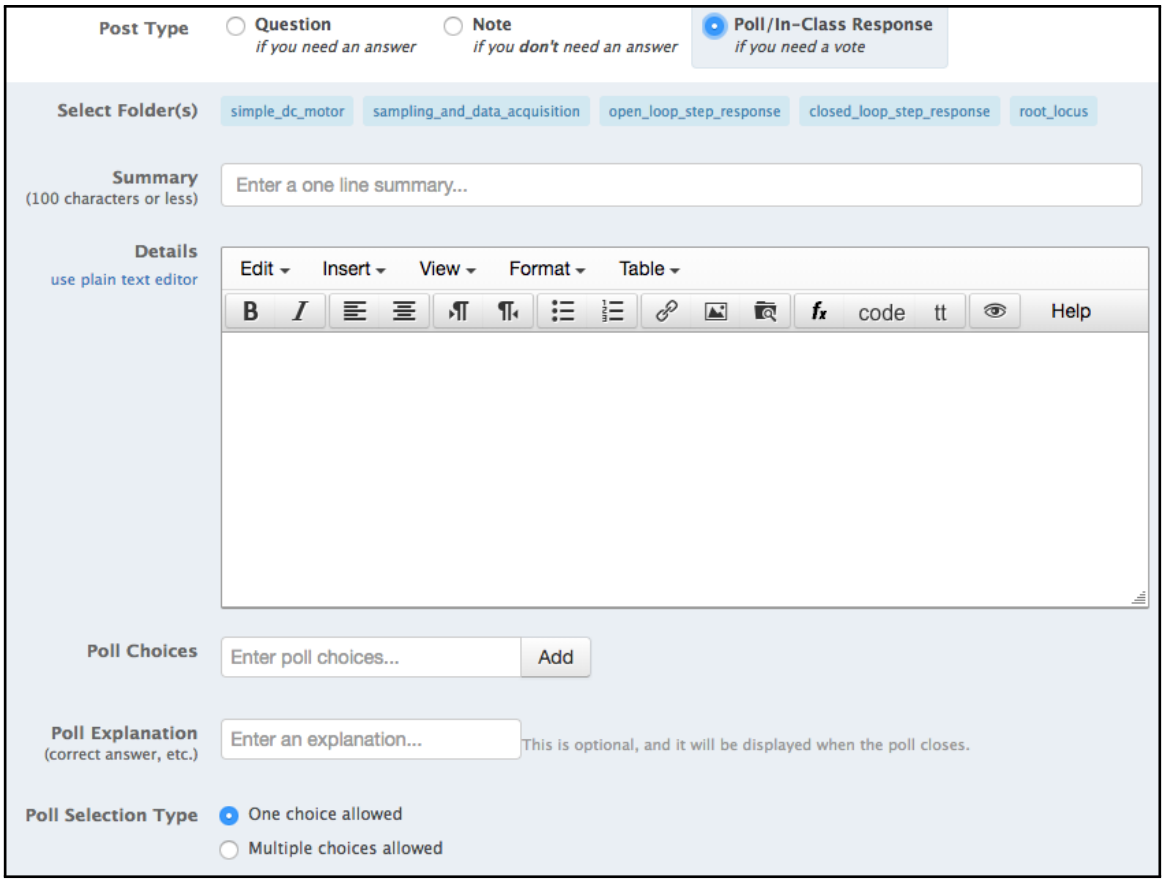



Figure 2.4: A New Poll/In-Class Response

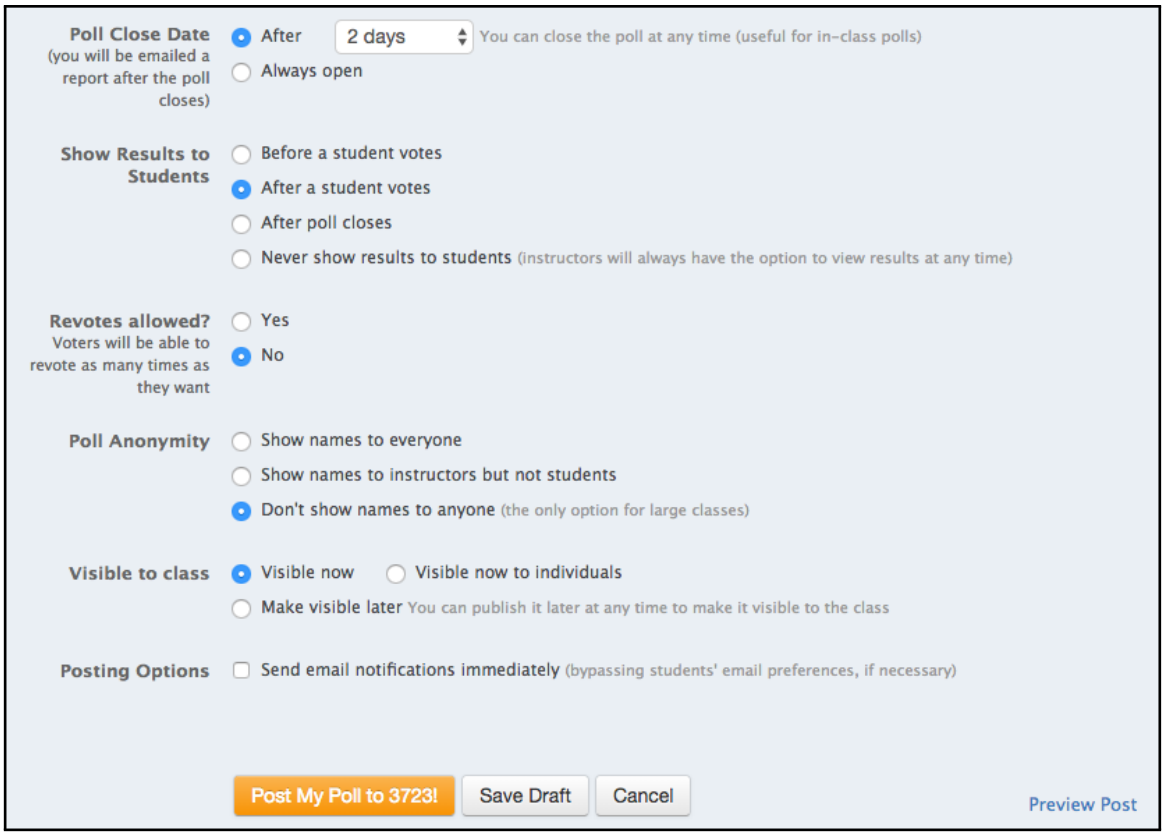



Figure 2.5: A New Poll/In-Class Response Cont.

When a new post is made, it shows up on the left in a preview bar that can be clicked on to answer (Figure 2.6). Figure 2.7 shows an example, from the instructor/TA point of view, of what the results show when a poll has been made and has closed.

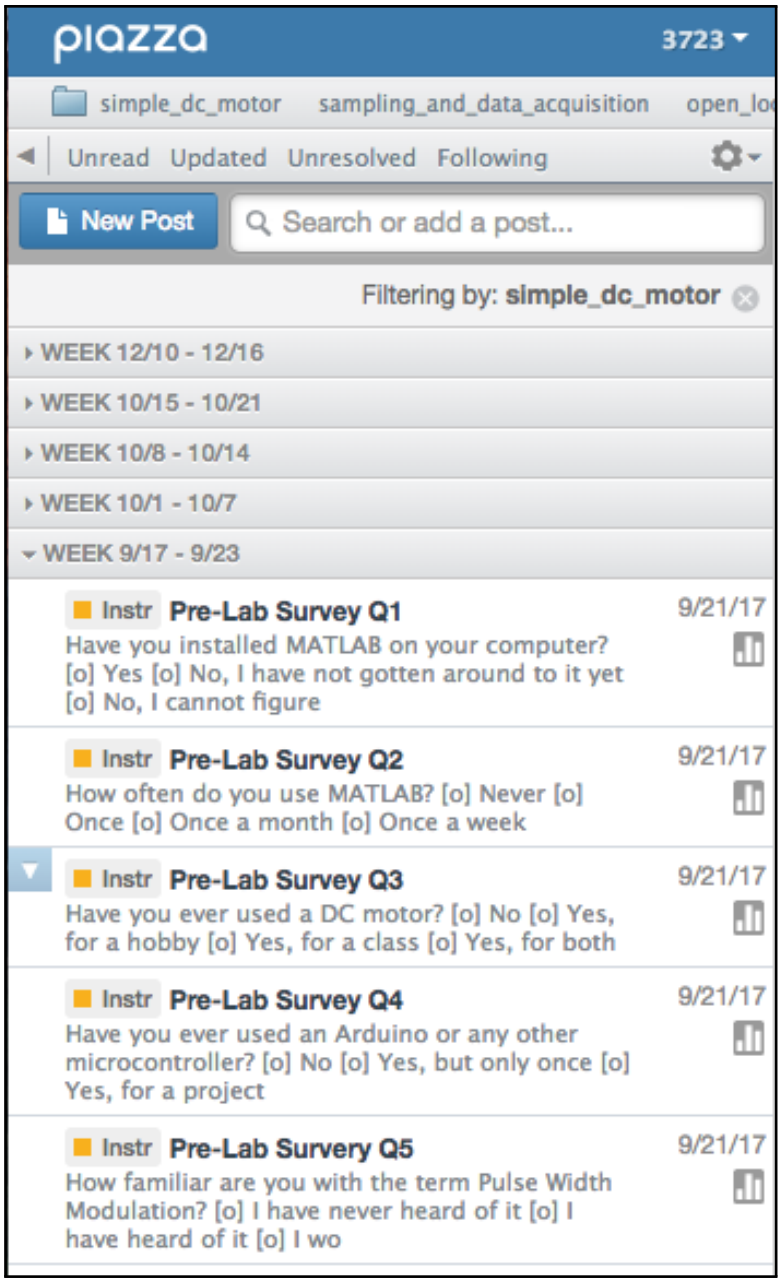



Figure 2.6: Example of Preview to Posts

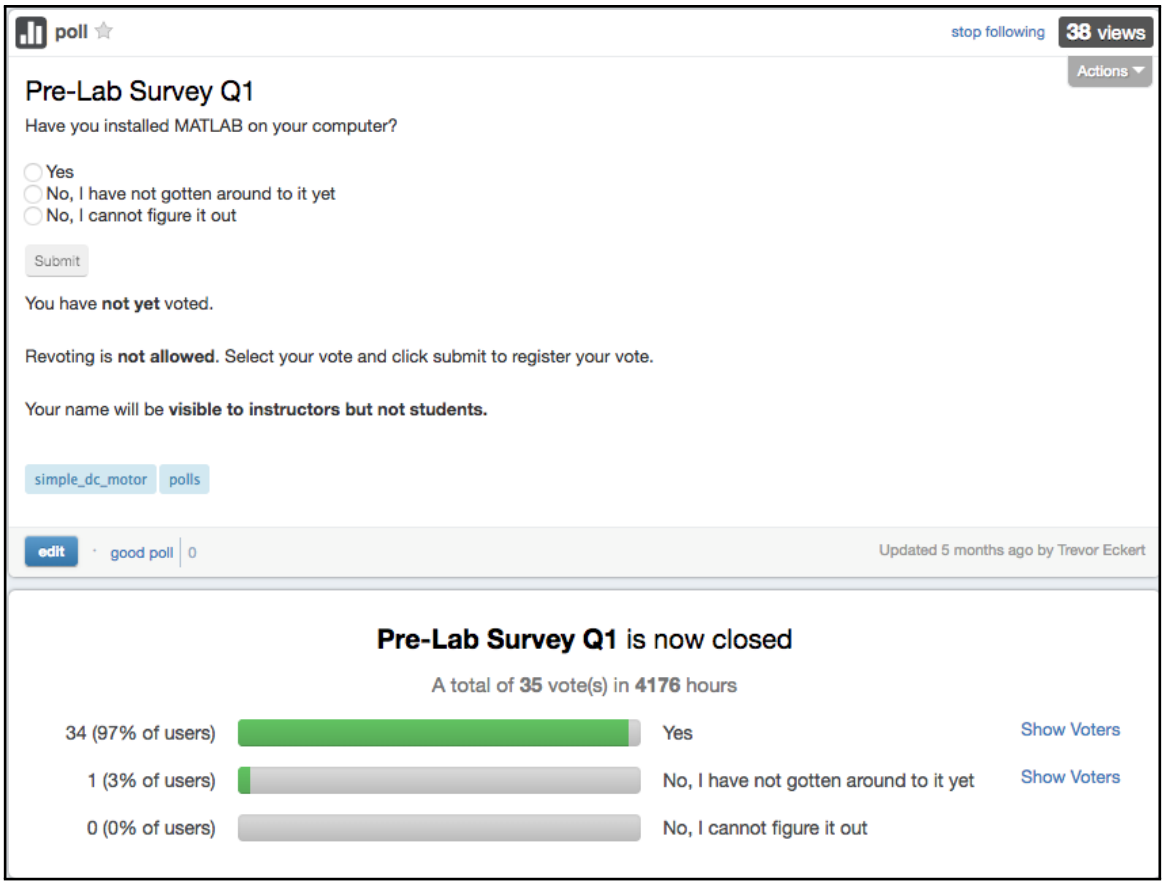



Figure 2.7: Example of Poll Results

### 2.1.2 Resources

The Resources tab can be useful to try and centralize all of the information about the course. Most universities use some form of online classroom to post key descriptions and documents about a class. Given that some students may have difficulty navigating through multiple websites, most everything that a typical online classroom can do can also be posted to the Resources tab of Piazza. Staff information can also be added to the Resources tab, like office hours and location (Figure 2.8). Other resources, such as the syllabus, homework, and project guidelines, can be posted as well (Figure 2.9). The only thing that Piazza does not have, in comparison with an online classroom, is a dropbox where students can submit project reports.

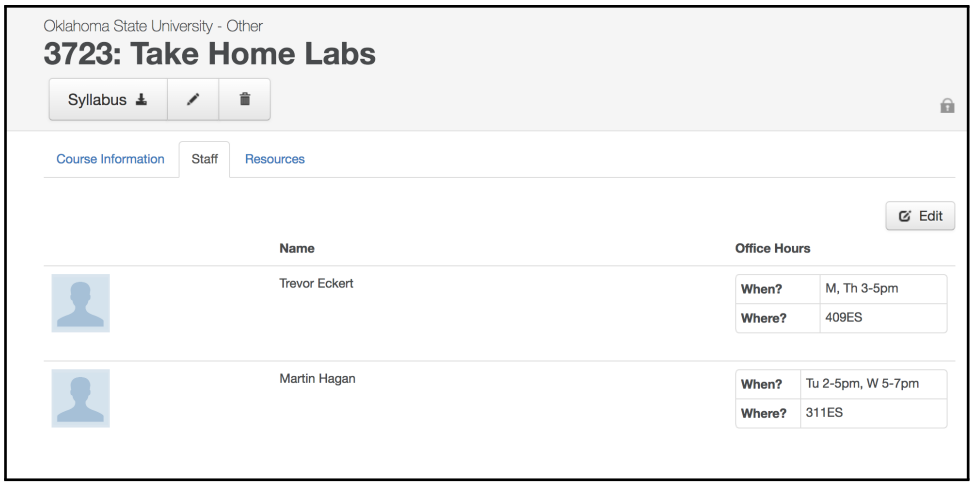



Figure 2.8: Staff Resources Page

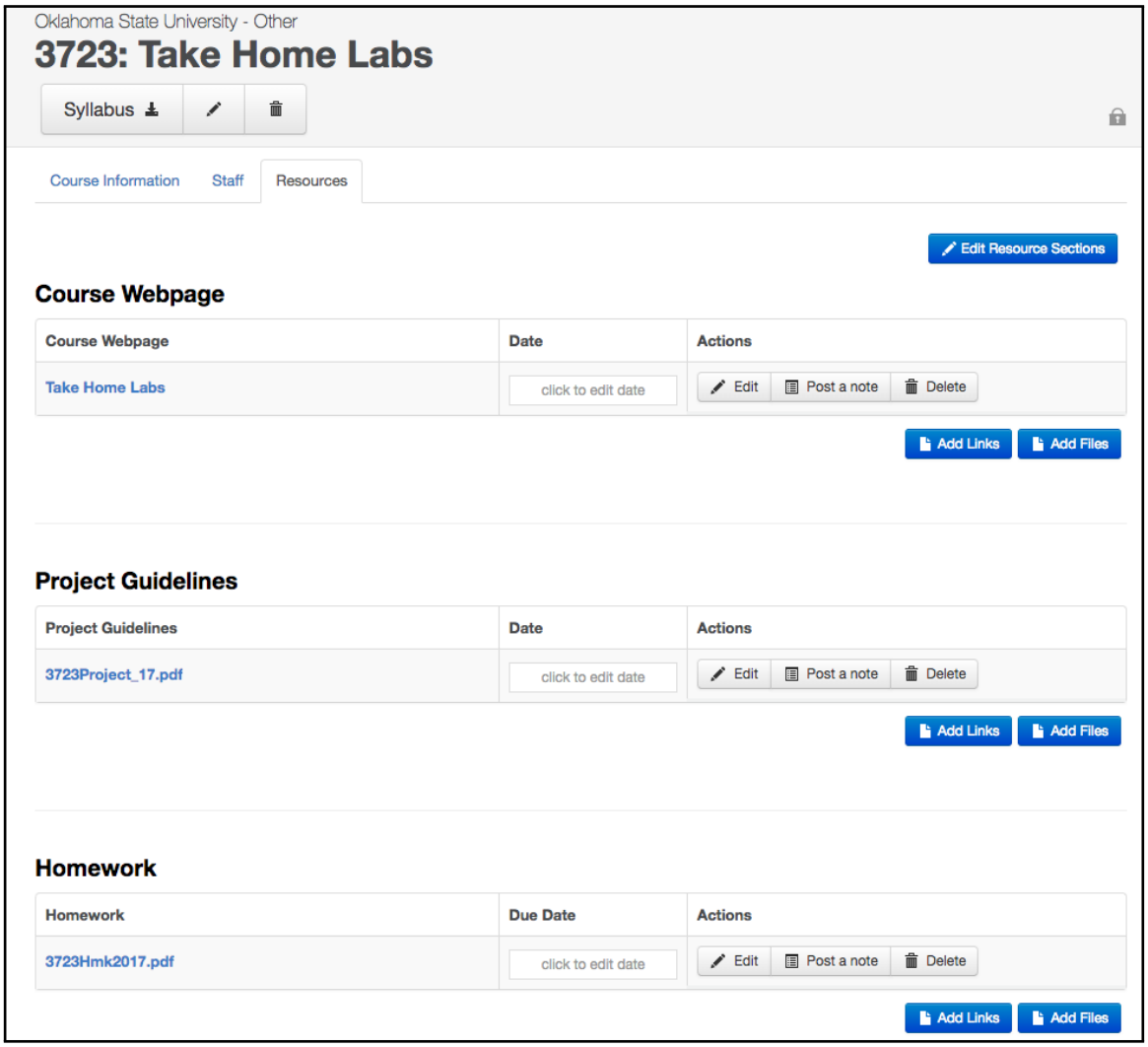



Figure 2.9: Other Resources Page

### 2.1.3 Statistics

The Statistics tab lets you look at all of the class statistics in-depth. The unique users per day gives an insight into the amount of activity among the students. Some other stats include: general class stats, top student contributors, top student askers, top student answerers, top listeners, and instructor and student participation. A CSV can also be downloaded that gives a statistic on every possible post and poll for each student if more in depth analysis is necessary.

### 2.1.4 Manage Class

The Manage Class tab covers the overall settings of the class. There are settings for Question and Answer that enable and disable anonymous posting, private posts, student polls, instructor tagging, and more. There is a section to customize your folders. There is also sections enabling enrollment of students, professor, and TAs. The Manage Class tab is the main hub to edit certain aspects of the Piazza class page.

## 2.2 Assessment

This section discusses the assessment used on the 2015 and 2017 labs. There are pre-lab and post-lab surveys for each 2017 lab, to gauge student progress on important topics. The students also give a free response answer to gauge any possible technical issues. Comparisons will be made between the 2015 and 2017 lab reports to assess the overall quality, to determine if any changes made from 2015 to 2017 were effective, and to find out if there are still any problems. Piazza will be used to assist with the assessment and to encourage student interaction.

A common assessment practice in education is Likert scale survey questions to gauge student confidence in having learned the material. The Likert scale is used to measure

attitudes in the form of five responses - Strongly Disagree, Disagree, Neither Agree Nor Disagree, Agree, and Strongly Agree. The advantage of using Likert scale opinion surveys is obtaining quantifiable data. A disadvantage of using opinion surveys is that the answers are not always true (thinking that you understand is not the same as understanding). Piazza will be used to pose a series of pre-lab and post-lab Likert scale survey questions to assess the attitude before and after each experiment (Figure 2.10). The responses will be anonymous to students to prevent as much social pressure as possible.

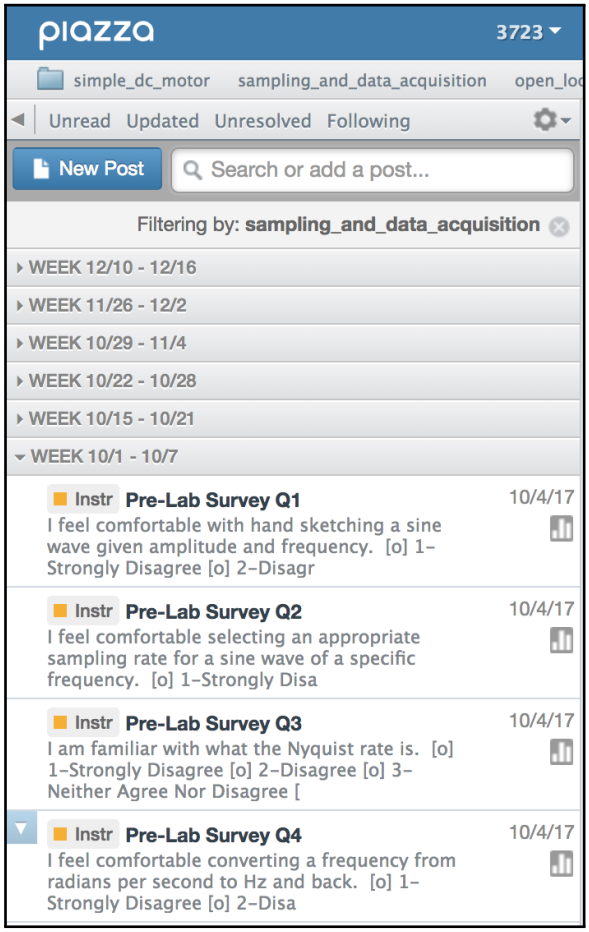



Figure 2.10: Example of Pre-Lab Survey For Sampling and Data Acquisition

Proper assessment should not stop at just the Likert scale survey questions. More information can be gained from student reports. Assessing how well students are able to convey their understanding of the theory, how they are able to make and comparisons

among theory, simulation, and experiment, is best done by a careful review of the lab reports. Comparison of the survey questions (primarily post-lab) with the student reports from 2017 will help assess the truthfulness of the survey answers and give a deeper understanding of unquantifiable aspects of THL.

Since in 2015 there were some issues, it will be useful to compare the 2015 and 2017 student lab reports. The changes that were made to counteract these issues in 2015 will hopefully be made evident in the 2017 student reports. In addition, the free response questions on Piazza will give students a chance to talk about what they like and do not like in every lab. This will help us assess if the changes worked and may suggest changes to be made for the future. Details of the assessment for each individual lab are in Chapters 3-7, and an overview of the key highlights of assessment is in Chapter 8.

# CHAPTER 3

# SIMPLE DC MOTOR

## 3.1 Objective

The Simple DC Motor experiment is a revision based on Sean Hendrix′s thesis [2]. It is an introductory experiment designed to help students get familiar with the hardware and some of the software used in later experiments. The students will learn how to successfully upload Simulink models to an Arduino Mega 2560 and get a basic understanding of the effect of different voltages applied to a DC motor via Pulse Width Modulation. With this comes some experience with static friction.

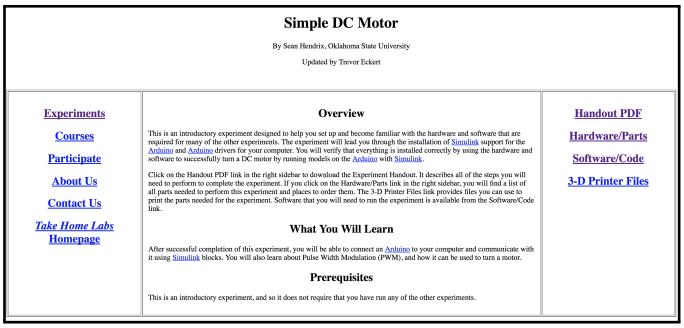



Figure 3.1: Simple DC Motor Webpage

## 3.2 Experiment Flow

This section will go over the basic flow of the experiment, but will not go into great detail for each step. Please refer to the project handout on the Take Home Labs website for more specifics. The experiment handout can also be found in Appendix E. Figure 3.1 shows the Simple DC Motor webpage.

### 3.2.1 Hardware Setup

This experiment starts off by explaining how to assemble all of the necessary hardware in order to perform many of the experiments that follow. It is assumed that the student has purchased and 3-D printed all of the required materials, which are listed under "Hardware/Parts" and "3-D Printer Files," respectively, on the Take Home Labs website under the Simple DC Motor Experiment page (see Figure 3.1).

### 3.2.2 Software Setup

Once the hardware is set up, the next step is to set up the required software. First, MATLAB and Simulink are required (all of the experiment revisions were made with MATLAB/Simulink 2017a). Then, the student is prompted to download a MATLAB script and Simulink model that is already created on the Take Home Labs website under "Software/Code". This will be used in the Pulse Width Modulation section of the experiment. Lastly, the students are prompted to install the "Simulink Support Package for Arduino Hardware," which can be installed through the "support package installer" in MATLAB.

### 3.2.3 Creating a Basic Simulink Model

Creating a Simulink model and/or uploading the model to an Arduino may be a foreign procedure to students. There are very specific details that must be followed in order to successfully perform this task. These tasks include

- Creating a blank Simulink model
- Dragging in blocks that are important to the laboratory
- Defining the solver and the fixed step size
- Setting up the model to run on the specific hardware

### 3.2.4 Pulse Width Modulation

Given a fixed DC power supply along with a digital output, it is not entirely clear how to change the voltage applied to the motor. This section of the experiment is an interactive exercise that gives a brief overview of Pulse Width Modulation (PWM), and explains how it is possible to alter the voltage that the DC motor sees using PWM. The students make a connection between the time response of the system and the frequency of the PWM signal by analyzing the script they were asked to download in the software setup section. This is described in more detail in Section 3.3.3.

### 3.2.5 Comparison of Duty Cycles

The last exercise asks the students to apply different voltages and describe what the motor is doing. They also learn to change the direction the motor turns. Another concept that appears while performing these operations is the introduction of static friction in motors. The students are asked to determine the minimum voltage that causes the motor to turn in each direction. This is a concept that the students need to understand in order to explain their results in future experiments.

## 3.3 Changes Made to the 2015 Experiment

This section discusses the changes made to the experiment, and the reasoning behind each change. The changes were made based on experiences during the System Dynamics class in the fall of 2015.

### 3.3.1 Selection of a New Motor

The motor originally used in the experiments (Mitsumi M25N-2R-14) started to become more difficult to find. It was difficult to match the cost, encoder resolution, and size of the original motor. Eventually, the Pololu 2821 motor, shown in Figure 3.2, was

selected. Pololu is a reputable company, and this is their largest motor with the highest encoder resolution. The motor should remain available for the foreseeable future. Table 3.1 shows some characteristics of each motor.

Figure 3.2: Pololu item #2821 Motor With Encoder

Table 3.1: Comparison Between Old and New Motor

| Characteristic | Mitsumi M25N-2R-14 | Pololu item #2821 |
|---|---|---|
| Cost | $26.31 | $24.95 |
| Encoder Resolution (CPR) | 1336 | 64 |
| Encoder Type | Optical-Quadrature | Magnetic-Quadrature |
| Maximum Voltage | 34V | 12V |
| Free-Run Speed | 11,000 rpm | 11,000 rpm |
| Free-Run Current | 100mA | 300mA |
| Weight | 66.7g | 110g |

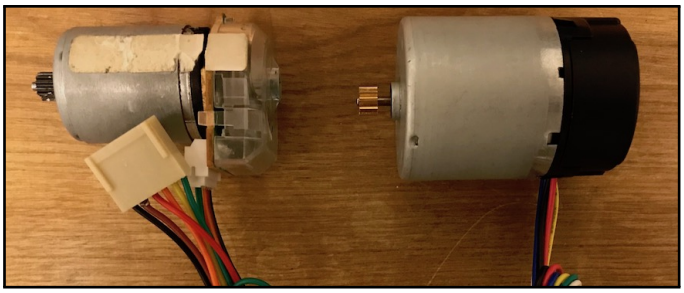


Figure 3.3: Old Motor and New Motor

One noticeable difference between the motors is the encoder resolution. Tests have shown that 64 counts per revolution is sufficient for our experiments, if a few small changes are made in the software. In addition, it makes sensor resolution a more obvious issue for the students to address.

### 3.3.2 Hardware Setup

The hardware setup section of the experiment changed significantly. In the previous Simple DC Motor experiment, the final hardware setup was not complete. Instead, the 3D printed parts were integrated into the Sampling and Data Acquisition lab. Now, there is a complete list of steps, which results in the final setup for future experiments. This change was made because the previous Sampling and Data Acquisition lab was found to be too long. Also, the previous Simple DC Motor experiment did not have too much material, so the swap made sense. The final hardware setup is shown in Figure 3.4.

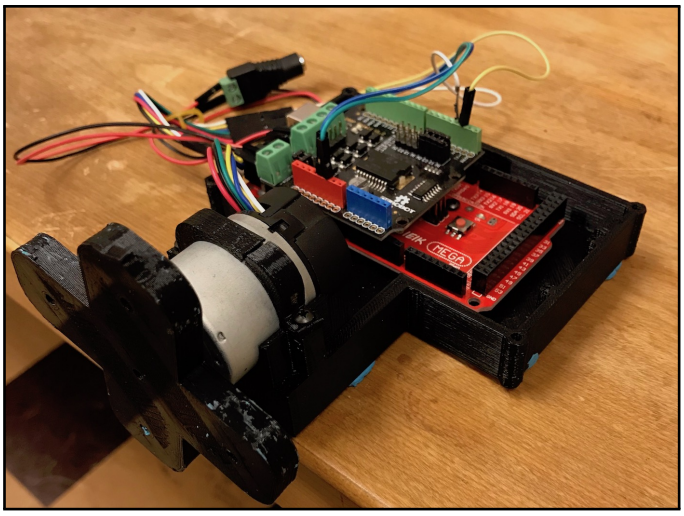


Figure 3.4: Total Lab Setup

### 3.3.3 Pulse Width Modulation

Given that the system uses a fixed 12V DC power supply and because the Arduino does not have any analog outputs, pulse width modulation (PWM) is used to vary the voltage applied to the motor. When the Take Home Labs were first integrated into the System Dynamics course, students had questions on how PWM worked. Thus a small PWM section was integrated into the Simple DC Motor lab. The following section is from the new handout, and also shows what results the students will obtain.

---

**Exercise 1: PWM**

Pulse Width Modulation (PWM) is a method for approximating the effect of an analog voltage by switching a constant (digital) voltage on and off quickly. PWM operation is defined by the percentage of time, called the duty cycle, that the signal is high during each period of the oscillation. Another name for the period is the cycle. For example, if a signal has a 50% duty cycle, then it is on for half of the period and off for half of the period. If the duty cycle is 25%, then the signal is on for 25% of the period and off for 75% of the period. For the Arduino, the frequency of this signal is 490 Hz, or period T =

$\frac{1}{490}$ seconds. The Arduino output receives an 8 bit digital value, which in decimal represents values 0 to 255. The value 0 represents 0% duty cycle, and 255 represents 100% duty cycle.

The average voltage generated by a PWM wave with amplitude $A$ can be calculated as:

$$s_{avg}(t) = \frac{1}{T}\int_0^T s(\tau)d\tau$$

*For the 50% duty cycle case*

$$s_{avg}(t) = \frac{1}{T}\Big[\int_0^{0.5T} Adt + \int_{0.5T}^{T} 0dt\Big]$$

$$s_{avg}(t) = \frac{1}{T}\Big[At\Big]\Big|_0^{0.5T}$$

$$s_{avg}(t) = \frac{1}{T}\Big[0.5AT\Big]$$

$$s_{avg}(t) = 0.5A$$

Figure 3.5 shows what a 50% and a 25% duty cycle square wave should look like.

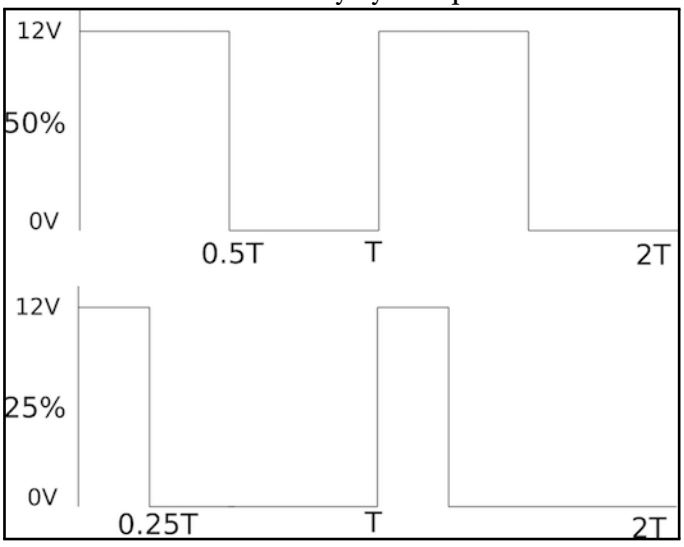



Figure 3.5: Example of PWM 25% and 50% Duty Cycle

A motor can be viewed as a lowpass filter (for example, the simple RC circuit shown in Figure 3.6), as you will see in later experiments. If the period of a PWM signal applied to a low pass filter is much shorter than the response time of the filter, then the capacitor won't have enough time to fully charge during one period. You will experiment with this

concept in the following steps.

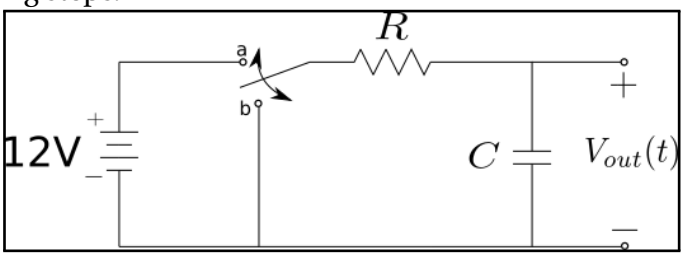



Figure 3.6: Simple RC Circuit

1. Derive a transfer function from the input voltage to the output voltage when $R = 1\Omega$ and $C = 1F$. The Simulink file *rc.slx* that you downloaded and extracted simulates this transfer function. It will be called by the file *rc_script.m* to simulate the operation with different PWM frequencies in the following steps.

   - Call the 12V input $V_{in}(t)$

$$v_{in}(t) - Ri - v_{out}(t) = 0$$

$$i = C\frac{dv_{out}}{dt}$$

$$v_{in}(t) - RC\frac{dv_{out}}{dt} - v_{out}(t) = 0$$

   Take the Laplace transform of the equation (0 initial conditions)

$$V_{in}(s) - RCsV_{out}(s) - V_{out}(s) = 0$$

$$V_{in}(s) - V_{out}(s)\left[RCs + 1\right] = 0$$

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{RCs+1}$$

   When R=1 and C=1

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{s+1}$$

2. From the Matlab command window, open the file *rc_script.m* that you downloaded and extracted earlier. It should open in the Matlab editor.

3. Run the *rc_script.m* file. In the editor, you can click the green run arrow. **Note:** three figures will pop up when the file has completed running.

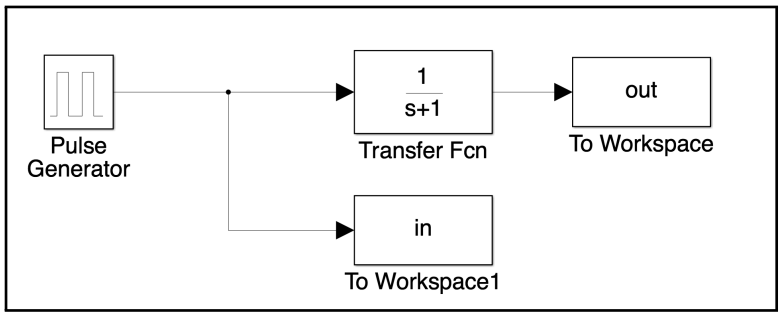



Figure 3.7: RC Simulation

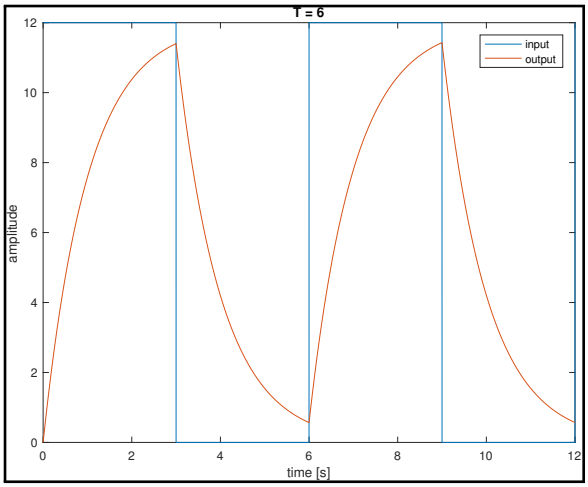



Figure 3.8: Simulation T = 6 seconds

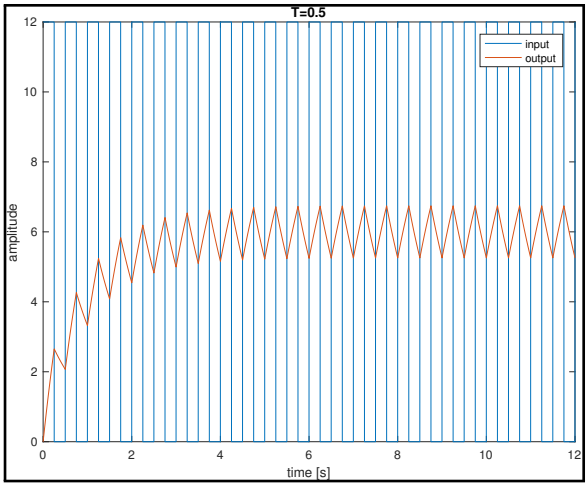



Figure 3.9: Simulation T = 0.5 seconds

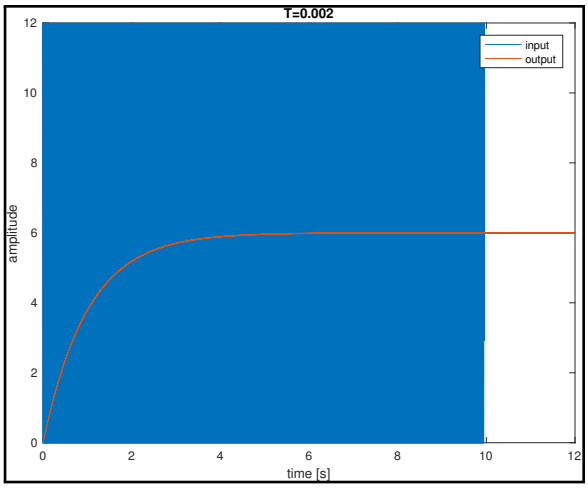



Figure 3.10: Simulation T = 0.002 seconds

4. Figure 3.8 shows a slow PWM frequency (6 sec period), Figure 3.9 shows a higher frequency (0.5 sec period), while Figure 3.10 shows a frequency similar to what the Arduino uses for a PWM cycle (0.002 sec period). Examine the three figures to get an understanding of why we need a fast PWM frequency if we want the response to the PWM signal to be similar to the response of a fixed voltage. (In other words, we want the response to a 50% duty cycle PWM signal, with 12 volt maximum, to be the same as a response to a constant 6 volt signal.)

5. Using the transfer function you found earlier, find the response to a 6 volt step function input and plot the response. Compare it to the plot in Figure 3.10 above. Is the response of the 50% duty cycle 12 volt PWM signal the same as the response to a 6 volt step function?

   - Define $v_{in}(t) = A \Rightarrow V_{in}(s) = \frac{A}{s}$

$$V_{out}(s) = \frac{1}{s+1}\frac{A}{s} = \frac{R_1}{s} + \frac{R_2}{s+1}$$

$$V_{out}(s) = \frac{A}{s} - \frac{A}{s+1}$$

$$v_{out}(t) = A - Ae^{-t}$$

$$v_{out}(t) = A\left[1 - e^{-t}\right]$$

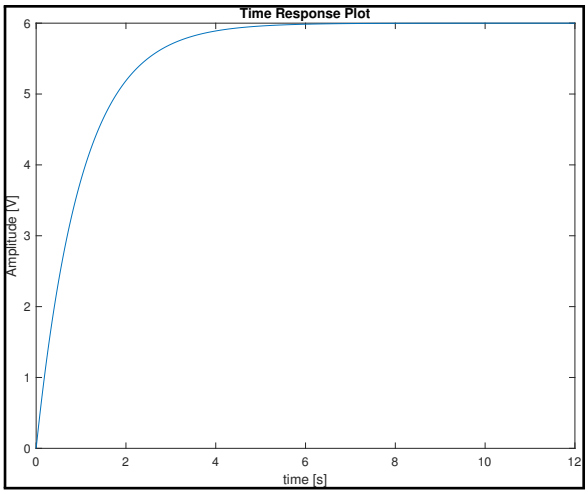



Figure 3.11: Time Response Plot

Figure 3.11 shows the plot generated in time using the derived $v_{out}(t)$ equation. Figure 3.10 and 3.11 are very similar.

6. Figure 3.13 shows how the 8-bit digital PWM code (from 0 to 255) is computed. Fill in Table 3.2 to develop an understanding of the resolution achieved by this process. For each voltage, compute the corresponding duty cycle of the PWM wave (assuming a 12 volt maximum). Then compute the corresponding digital value (from 0 to 255) that will produce the desired duty cycle. In addition, discuss the minimum change in duty cycle that you can acheive with the 8 bit digital code (0 to 255). You will add the description of the motor operation in the next exercise.

Table 3.2: Voltage Table

| Voltage | Duty Cycle (%) | Digital Value$_{10}$ | Describe Motor Operation |
|---|---|---|---|
| 0.5 | ***4.31*** | ***11*** | |
| 1 | ***8.24*** | ***21*** | |
| 1.5 | ***12.55*** | ***32*** | |
| 2 | ***16.86*** | ***43*** | |
| 4 | ***33.33*** | ***85*** | |
| 6 | ***50.20*** | ***128*** | |
| 8 | ***66.67*** | ***170*** | |

When the students fill out the table in regards to the Duty Cycle and the Digital value representation, they should realize what quantization means. For instance, 6 volts should represent 127.5 but that value cannot be represented in 8 bits. Instead the value is quantized to 128, thus resulting in a quantization error giving a 50.2% duty cycle.
In the "Describe Motor Operation" column, the motor should not move 0.5 volts due to friction. Then the speed of the motor should increase as the voltage is increased. The addition of this exercise gives the students insight to how PWM works on a DC motor.

---

### 3.3.4 Simulink Block Diagram Arrangement

Given the new PWM section of the experiment, it seemed appropriate to arrange the hardware blocks to clearly show the conversion of voltage to a digital value that represents the duty cycle. The original layout for the Simple DC Motor lab, as shown in Figure 3.12, lumped the conversion into one gain block. Figure 3.13 shows each step: 1) voltage being divided by the max motor voltage and 2) the ratio being scaled from 0 to 255. The 8-bit value is what the PWM block uses in order to send the duty cycle regulated voltage

to the motor.

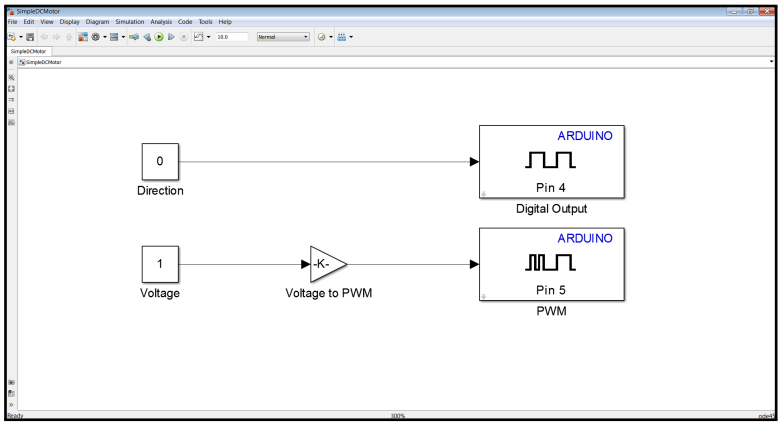



Figure 3.12: Previous Simulink Block

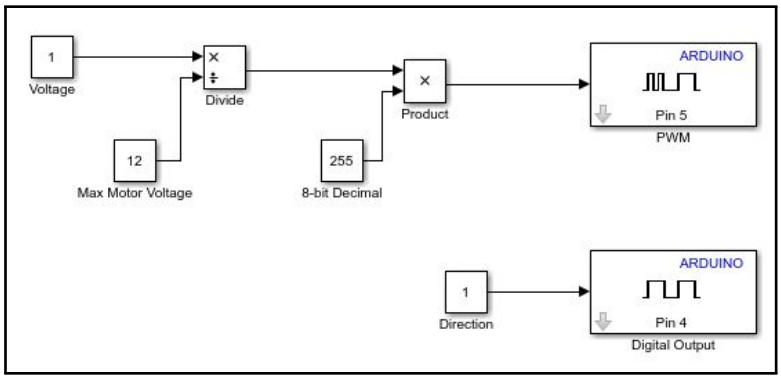



Figure 3.13: Final Simulink Model

There are two main benefits to this change:

1. It gives the students a better understanding of what the block diagram is doing.

2. It gives the students more familiarity with Simulink, and it reduces the setup time for future experiments.

### 3.3.5 Elimination of External Mode

The original format of the Simple DC Motor lab had an exercise of making observations of the motor in something called external mode. In external mode the code could be updated in real time when it came to adjusting values (i.e. motor voltage), instead of uploading the Simulink model to the board every time a value was changed. Taking out external mode and keeping only normal mode made the lab shorter, which made it

possible to have the students focus on the main objectives. The pulse width modulation section was made possible because of the elimination of external mode. External mode made up an even larger part of the Sampling and Data Acquisition lab, which will be discussed in the next chapter.

### 3.3.6 Table of Discussions and Questions

When students perform experiments at home, they tend to skip steps and not read the handout in its entirety. When the labs were used in 2015, many students failed to answer questions and were very brief in their discussion sections. To ensure that a question or discussion is not overlooked, a table was added at the end of the new handouts to remind the students to go back and check their work. The table for the Simple DC Motor follows:

---

**Table of Discussions and Questions**

Before you turn in your report for this experiment, make sure that you have answered all of the questions that have been posed. It is important that your answers be expansive and that they demonstrate that you were mentally engaged in the experiment. Below is a recap of the important questions and the number of the step where each question was embedded.

| steps | Discussion/Question |
|---|---|
| 61 | Transfer function of RC network |
| 64 | Discussion on different PWM frequencies |
| 65 | Comparison of theoretical step response and PWM response |
| 66 | Voltage resolution due to digital representation |
| 68 | Fill out the voltage table |
| 69 | Descriptions of motor operation |
| 72 | How does direction line affect motor operation? |
| 73 | Friction threshold for each direction |

### 3.4 Assessment and Results

This section will go over the Piazza surveys that were posted before and after the experiment, make connections to what students feel they have improved on, and tie it all together through the students reports as well as reports from the Fall of 2015 Systems Dynamics course.

#### 3.4.1 Pre-Lab Questions

Before the students preformed the Simple DC Motor lab experiment, five questions were posed to get a feel for the students general prior knowledge when it comes to MATLAB and Simulink, DC motors, microcontrollers, and pulse width modulation. The questions were asked in a multiple choice format as follows:

1. Have you installed MATLAB on your computer?

    (a) No, I cannot figure it out.

    (b) No, I have not gotten around to it yet.

    (c) Yes.

2. How often do you use MATLAB?

    (a) Never.

    (b) Once.

    (c) Once a month.

    (d) Once a week.

3. Have you ever used a DC motor?

    (a) No.

    (b) Yes, for a hobby.

    (c) Yes, for a class.

    (d) Yes, for both.

4. Have you ever used an Arduino or any other microcontroller?

    (a) No.

    (b) Yes, but only once.

    (c) Yes, for a project.

5. How familiar are you with the term Pulse Width Modulation?

    (a) I have never heard of it.

    (b) I have heard of it.

    (c) I would be able to explain it partially.

    (d) I would be comfortable explaining it to someone else.

The results of these questions are shown in Table 3.3.

Table 3.3: Pre-Lab Simple DC Motor Results

| Question | (a) | (b) | (c) | (d) |
|---|---|---|---|---|
| Have you installed MATLAB on your computer? | 0% | 3% | 97% | N/A |
| How often do you use MATLAB? | 9% | 24% | 44% | 23% |
| Have you ever used a DC motor? | 26% | 9% | 53% | 12% |
| Have you ever used an Arduino or any other microcontroller? | 21% | 21% | 58% | N/A |
| How familiar are you with the term Pulse Width Modulation? | 29% | 32% | 24% | 15% |

### 3.4.2 Post-Lab Questions

After students completed the lab, additional questions were posed. This allowed us to make before and after comparisons. The questions were based on a Likert scale and are as follows:

1. I feel comfortable navigating around in MATLAB and Simulink.
   (a) Strongly Disagree
   (b) Disagree
   (c) Neither Agree Nor Disagree
   (d) Agree
   (e) Strongly Agree
2. I feel comfortable interfacing an Arduino with Simulink.
3. I feel comfortable turning a DC motor in both directions.
4. I feel comfortable with how Pulse Width Modulation works.

The results of the questions are shown in Table 3.4.

Table 3.4: Post-Lab Simple DC Motor Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I feel comfortable navigating around in MATLAB and Simulink. | 0% | 0% | 12% | 67% | 21% |
| I feel comfortable interfacing an Arduino with Simulink | 3% | 6% | 15% | 61% | 15% |
| I feel comfortable turning a DC motor in both directions. | 0% | 0% | 3% | 48% | 48% |
| I feel comfortable with how Pulse Width Modulation Works | 3% | 6% | 27% | 36% | 27% |

### 3.4.3 Comparison of Pre-Lab and Post-Lab Answers

In this section, complimentary pre-lab and post-lab questions will be compared to see how well the students progressed through the lab.

#### Comfort With MATLAB and Simulink

As shown in Table 3.3, most students have only used MATLAB once a month or less. This suggests that they are not entirely comfortable using MATLAB or even navigating through the software. Table 3.4 suggests that after just one lab experiment, the majority of the students are starting to feel comfortable navigating around in MATLAB and Simulink.

#### Interfacing Simulink with Arduino

Even though most students have at least used an Arduino or another type of microcontroller, uploading code to an Arduino via Simulink seemed to be a new concept for most, if not all students. After the lab was completed, 76% of the students claimed they felt comfortable with the process of running Simulink models on Arduino.

#### Manipulation of a DC Motor

Since this lab is an introductory experiment, getting the motor to turn at different speeds directions was all that was necessary. Before they preformed this experiment, it seemed like a good majority have at least used a DC motor in some way shape or form. After the lab, every single student except for one answered that they were at least comfortable. One student neither agreed nor disagreed.

#### Pulse Width Modulation

This concept seemed to be the most difficult for students to grasp. The objective of this section of the experiment was to give students a basic understanding of how to vary a voltage continuously when only a digital output is available. The pre-lab question tied to this concept shows that students had little prior knowledge of PWM. The objective of the lab was reached, since on the post lab survey the majority of the students agree or strongly agree that they were comfortable with PWM.

### 3.4.4 Observations of Lab Reports and Student Interactions

This section considers assessments in addition to the Piazza surveys: 1) a comparison of old lab reports (Fall of 2015) versus new reports (Fall of 2017), 2) a comparison of Piazza survey answers and lab reports (to check the honesty of the Piazza surveys), and 3) other observations from interactions with the students.

#### Old Reports Versus New Reports

Overall , the addition of the table of questions was effective. Although some students continued to skip questions, there was a noticeable improvement from 2015. Although it is difficult to get students to be expansive in their discussions, the 2017 reports showed improvement.

The main change in the 2017 handout was the addition of the PWM section. In

the 2015 reports some students got confused about how PWM worked. The original Simulink model made it difficult to see the process. Making the change showed promising results in the 2017 lab reports. In the 2015 reports students did not explain as much about how each applied voltage affected the velocity of the motor. The change for the new handout made it possible to see that students made the connection between voltage and velocity.

**Survey Questions Versus Lab Reports**

The majority of students show that they are comfortable navigating around in MATLAB and Simulink. Performing this experiment, one of the goals was to get the students familiar with the software. There was no indication in the lab reports that students had difficulty performing each step to set up the software that was deployed to the Arduino.

In the survey, some students claimed that they were not comfortable interfacing an Arduino with Simulink, but in order to properly execute the experiment they have to upload code several times to the Arduino. Although the majority did feel comfortable, the small percentage that said they did not raises some concerns for future experiments, even though every student was able to get the motor to operate.

Based on their reports, getting the motor to turn in both directions was easy for the students. This aligns nicely with the post-lab questions and held true to their results in their experiment reports.

The PWM exercise proved to be the most difficult section in terms of tying the theory to the experiment. There were a good number of students who stood neutral on this topic. Almost every student showed that they were able to derive the transfer function of the RC circuit as well as to at least discuss a little bit about the three figures from the

MATLAB script. The student discussions proved that they have some understanding of PWM, but maybe not enough to feel completely comfortable.

#### Other Observations

Most of the time, when a student came for help during the semester, they had usually skipped a step or did not read the instructions in their entirety. Another thing students noticed is that when they first plugged in the motor, it made a noise but did not move. This is because the first voltage they apply is 0.5 Volts, which is usually below the static friction threshold. Some students stopped there, because they thought something was wrong. Most students were able to upload the Simulink model to the board properly and apply different voltages to the motor.

### 3.5 Conclusion

Given that this is an introductory lab, there is not much material covered. The goals were: 1) to get students familiar with some of the Simulink toolboxes for Arduino, 2) to have them learn to turn a DC motor, 3) and to demonstrate how it is possible to vary voltages continuously with a digital output. All-in-all, there did not seem to be many issues for the students, and the post-lab survey questions proved to be almost totally consistent with their discussions and answers in their experiment reports. The changes made to the experiment from 2015 had the intended effects.

# CHAPTER 4

# SAMPLING AND DATA ACQUISITION

## 4.1 Objective

The Sampling and Data Acquisition experiment is a revision based on Carion Pelton′s thesis [1]. It is also an introductory experiment that is designed to help students understand the concept of sampling and how that relates to acquiring data with the encoder in real time. Multiple sampling rates are chosen through simulation as well as on the physical system to reinforce the concept of sampling and the limitations of the physical system.

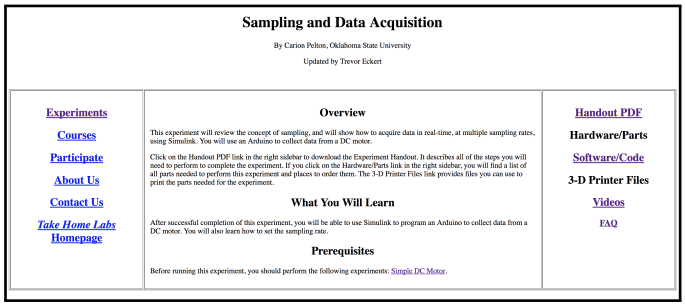



Figure 4.1: Sampling and Data Acquisition Webpage

## 4.2 Experiment Flow

This section will go over the basic flow of the experiment, but will not go into great detail for each step. It will include how to setup the software, sampling theory, sampling simulation, and a sampling experiment. Making the connection between all three sections of the lab - theory, simulation and experiment is - of key importance in this lab, as well as within all of the labs. Please refer to the project handout on the Take Home Labs website for more specifics. The experiment handout can also be found in Appendix F.

Figure 4.1 shows the Sampling and Data Acquisition webpage.

### 4.2.1 Software Setup

Since there is no hardware setup, the first thing to do in this experiment is to setup some of the software used to collect data from the encoder. Some of the Simulink blocks used in the Take Home Labs experiments are in a library referred to as the THL library (THLlib) [1] [2]. This experiment uses two pieces of software from the THL library: 1) the encoder block, which is used to collect data from the encoder and 2) the plot data single block, which collects data to plot in real time. The plot data single block will also send the data to the MATLAB workspace.

### 4.2.2 Sampling Theory

The first exercise of this experiment gives a brief overview of sampling theory and the Nyquist Theorem. The students are then asked to hand sketch a sine wave of a given amplitude, frequency, and phase. After they hand sketch the continuous sine wave, they are asked to draw the same wave form sampled at various sampling intervals.

### 4.2.3 Sampling Simulation

Following the hand sketching exercise, the students are prompted to simulate and plot sampled sine waves and to compare the plots with the hand sketches to reinforce sampling concepts.

### 4.2.4 Sampling Experiment

In this section of the lab the students use Simulink software to perform a physical sampling experiment. Since the motor is linear, if the input to the motor is a sine wave of a given frequency, the output will be a sine wave of the same frequency. In this experiment students apply a four volt amplitude sine wave with frequency of 1 Hz to the motor

and sample the velocity of the motor at several sampling intervals. They then make the comparisons between theory, simulation, and experiment.

## 4.3 Changes Made to the 2015 Experiment

This section discusses the changes made to the Sampling and Data Acquisition Lab, and the reasoning behind each change. The changes were made based on experiences during the System Dynamics class in the Fall of 2015.

### 4.3.1 Elimination of External Mode

Originally, one of the key concepts covered was the difference between the two methods for executing Simulink code on the Arduino: normal and external mode. It seemed like a good idea in the beginning to include this idea, because the sampling characteristics of the two methods are very different. However, it distracted students from the main goal (the concept of sampling). Also, there were a large amount of complaints that the previous Sampling and Data Acquisition lab was too long. Given that both of these issues seemed to be significant, it was deemed necessary to get rid of external mode entirely.

### 4.3.2 Elimination of Position Measurement

In the previous Sampling and Data Acquisition lab, the position of the motor was plotted. The position data was then differenced to approximate the velocity. Students make comparisons of these two plots with their theory and simulation. It was observed that students got confused with the position measurement plot because of drift. An example of this plot is shown in Figure 4.2. Notice how the response drifts up. This occurred because the motors turned more easily in one direction. It appeared to be confusing for most students and they believed that something was wrong with the hardware and/or their software. Figure 4.3 shows an example of a motor velocity plot. There

is almost no drift. We decided that using only the velocity plot could potentially reduce confusion.

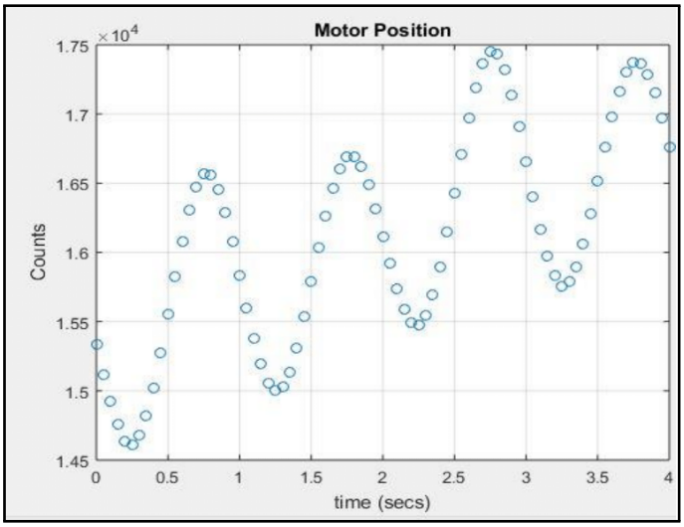



Figure 4.2: Motor Position Plot

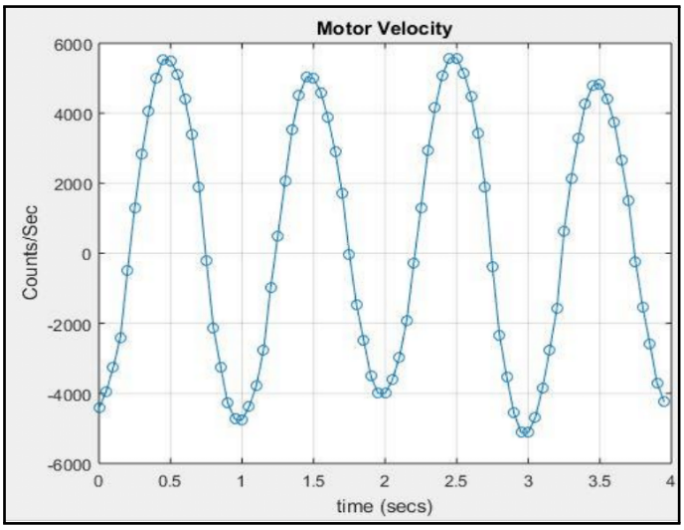



Figure 4.3: Motor Velocity Plot

### 4.3.3 Elimination of Byte Adjust

In the experimental section of this lab, when students apply a sine wave voltage to the motor, they use some software to read the serial port of the Arduino to collect encoder data to obtain and plot motor position. In 2015, we obtained this software from the Rensselaer Arduino Support Package Library (RASPlib). The software read the serial port one byte at a time, and then combined adjacent bytes to produce a single encoder

reading. In order to align the bytes so that the two bytes both corresponded to the same encoder reading, there was a "byte adjust" button in the plot window. If the bytes were mismatched, the plot would display unusual numbers. The user would then click the byte adjust button, and the software would shift one byte and recalculate the position. After a little experience, it was possible to get the plot to eventually produce the correct result, but it was somewhat tricky to use. In 2015 this became a major stumbling block for some students. Some claimed in their lab reports that they spent up to 6 hours to get the plot to work.

Because of these issues in 2015, we modified the serial plot software. We removed the byte adjust button, and, instead, adjusted the byte alignment in the software until the position calculation was reasonable. This simplified the use of the program significantly. Based on student feedback in 2017, which will be discussed in a later section, the new software worked quickly and easily for all students.

#### 4.3.4 Rearrangement of Sampling Intervals

In the previous Sampling and Data Acquisition lab, during the experimental section, students began with the lowest sampling rate and worked up to the highest rate. Unfortunately, at the lowest sampling rate it can be difficult to identify a sine wave. This caused some students to question their setups and to stop performing the experiment. In a standard lab, a student could check with the TA. With Take Home Labs, we decided to start with the highest frequency where the sine wave is clear.

#### 4.3.5 Reorganization of Handout

In this experiment, students will hand sketch a sine wave and then hand sketch a sampled version of the sine wave at different sampling rates. Then they will simulate the same thing through Simulink. Lastly, they will experimentally collect data that rep-

resents the theory and simulations. Once all of that is done, students will make comparison between all three sections in the lab. Comparison between theory, simulations and experiment is the overall theme in the Take Home Labs, and therefore the same amount of emphasis should be put on all three sections of the Sampling and Data Acquisition lab.

In 2015, a good amount of students left out all of the hand sketches in their reports. That eliminates the point of a theory section if students just skip it. This will also cause students to not draw the connections among theory, simulation, and experiment. In order to ensure that the students preform all three sections in the lab, we wanted to isolate all three sections from one another.

In the previous Sampling and Data Acquisition lab handout, the three sections of the lab were not isolated enough to prevent students from skipping ahead. Instead, the handout seemed to flow into each section of the lab without getting the students to stop and reflect on each important component. Two things were done to get more students to include their hand sketches: 1) adding a note to remind them to make sure they have done this before moving on and warning them that it is crucial to have the hand sketches done and 2) instead of having too many steps, which students are likely to skip, making a table that lists everything they need to do. Table 4.1 shows the inserted table that replaced about 15 steps in the 2015 handout.

Table 4.1: Sine Waves to Sketch and Simulate

| Sine Wave Frequency f | Sine Wave Period T | Sampling Frequency $f_s$ | Sampling Interval $T_s$ |
|---|---|---|---|
| 1 Hz | 1 second | 16 Hz | 0.0625 seconds |
| 1 Hz | 1 second | 8 Hz | 0.125 seconds |
| 1 Hz | 1 second | 4 Hz | 0.25 seconds |
| 1 Hz | 1 second | 2 Hz | 0.5 seconds |

## 4.4 Assessment and Results

This section will go over the Piazza surveys that were posted before and after the experiment, identify what topics students feel they have improved on, and tie it all together through the student reports from 2017 as well as reports from the Fall of 2015.

### 4.4.1 Pre-Lab Questions

Before the students preformed the Sampling and Data Acquisition lab experiment, four questions were posed to get a feel for students general prior knowledge when it comes to sketching sine waves, picking appropriate sampling rates, the Nyquist Theorem, and frequency conversions. The questions were asked in the form of a Likert scale as follows:

1. I feel comfortable with hand sketching a sine wave given amplitude and frequency.
    (a) Strongly Disagree
    (b) Disagree
    (c) Neither Agree Nor Disagree
    (d) Agree
    (e) Strongly Agree

2. I feel comfortable selecting an appropriate sampling rate for a sine wave of a specific frequency.

3. I am familiar with what the Nyquist rate is.

4. I feel comfortable converting a frequency from radians per second to Hz and back.

The results of these questions are shown in Table 4.2.

Table 4.2: Pre-Lab Sampling and Data Acquisition Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I feel comfortable with hand sketching a sine wave. | 3% | 6% | 10% | 45% | 35% |
| I feel comfortable selecting an appropriate sampling rate | 3% | 13% | 32% | 42% | 10% |
| I am familiar with what the Nyquist rate is | 45% | 29% | 16% | 6% | 3% |
| I feel comfortable converting from radians per second to Hz and back | 3% | 3% | 16% | 35% | 42% |

Table 4.2 shows that 80% of students feel comfortable hand sketching a sine wave. This is a topic that students should not have any problem with, especially since this is a junior level class. The honesty of these answers will be tested when we assess the student lab reports.

Something interesting is shown in Table 4.2 - in question two, 52% of students expressed comfort in choosing sampling rates, but in question three only 9% of students were familiar with the Nyquist rate. The answers to these two pre-lab questions contradict each other, which suggests that many students may be over confident in their knowledge of sampling.

As for the last question in Table 4.2, about 77% students claim that they can convert frequencies from radians per second to Hz and back. This claim will also be tested when we assess the lab reports.

### 4.4.2 Post-Lab Questions

After students completed the lab, additional questions were posed. This allowed us to make before and after comparisons. The questions were based on a Likert scale and are as follows.

1. I know how to use the Nyquist Theorem to choose a sampling rate.
2. I understand the difference between sampling frequency and the frequency of a sine wave.
3. I can find the period of a sine wave from the frequency.

The results of the questions are shown in Table 4.3.

Table 4.3: Post-Lab Sampling and Data Acquisition Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I know how to use the Nyquist Theorem to choose a sampling rate | 3% | 3% | 6% | 58% | 29% |
| Difference between sampling frequency and the frequency of a sine wave | 0% | 3% | 10% | 55% | 32% |
| I can find the period of a sine wave from the frequency | 0% | 0% | 3% | 42% | 55% |

Table 4.3 shows that after the students performed this lab, 87% of them now feel comfortable using the Nyquist Theorem to choose a sampling rate. This is a significant

increase over the 9% of the students who were familiar with the Nyquist Theorem before the lab.

The difference between sampling frequency and the frequency of a sine wave is a very important concept. 87% of students feel comfortable being able to understand the differences. However, this topic was a major issue in 2015. The purpose of this post-lab question was to enable us to compare students perceptions of their understanding of these concepts with their ability to use them, as evidenced in their lab reports, which will be discussed later.

97% of students claimed that they can find the period of a sine wave from the frequency. Another important thing to note is that 0% of students disagreed or strongly disagreed with the statement. Being able to understand the relationship between these conversions is an important concept that the students believe they have a grasp on, although we will test that belief when we assess the lab reports.

### 4.4.3 Comparison of Pre-Lab and Post-Lab Answers

In this section, complimentary pre-lab and post-lab questions will be compared to see how well the students progressed through the lab.

#### Choosing Sampling Rate via Nyquist Theorem

As stated earlier, there was a contradiction between the answers to two pre-lab questions. Since 52% of students said that they knew how to select an appropriate sampling rate, but only 9% were familiar with the Nyquist rate, an assumption was made that the 9% group is included in the 52% group. With that being said, the conclusion is that only 9% of students truly knew how to select an appropriate sampling rate. After the lab was performed, again, 87% of students stated that they now feel comfortable using the

Nyquist Theorem. This is a significant increase, which is one of the main goals of this experiment.

**Frequency Conversions**

In the pre-lab question about converting frequencies from radians per second to Hz and back and the post-lab question about finding the period of a sine wave from the frequency, 77% and 97% claimed comfort in these topics, respectively. While the change from 77% to 97% suggests a slight increase in confidence, these topics are easy to assess in greater detail via student lab reports.

### 4.4.4 Free Response Questions

Likert scale questions cannot elicit as much information as free response questions. Given the capabilities of Piazza, both are possible. We asked the students the free response questions "Describe which parts of the lab took the longest time and/or were cumbersome. Describe which parts of the lab were the most interesting." These questions would, hopefully, give us more insight into what the students thought about the experiment as a whole. This section of the thesis will go over some free response answers that either were common or appeared to be important.

One of the biggest time commitments some students had was installing the software to read the encoder and serial port (THLlib). However, other students didn't have any problems at all with installing or using the software. This may suggest that some students did not follow the instructions in their entirety. Some evidence for this is the fact that several students posted questions on Piazza about software problems they were having. In almost every case they had not followed some step that was written in the handout. All Mac users described having an issue with the software. A typical comment on this subject was "Getting through the Mac-specific bugs took the longest".

Overall, it did not appear that the time to download the software was a significant burden on most students. In fact, unlike 2015, no students complained about the time to do the lab. There were some extra steps that needed to be taken by Mac users. A potential fix for the MAC issue is to have a separate handout or setup FAQ for MAC users.

The other biggest time commitment students mentioned was "drawing the hand sketches". Even though this may seem like a problem, this is something we want the students to spend the most time on. The theory part should take the longest, especially if it is not something students are familiar with.

Most students stated that they liked seeing how the theory, simulation, and experiment all matched. They also enjoyed using the Arduino to turn the motor and display the motor position. Some students said that the lab was straightforward and easy to follow.

One other interesting quote from a student: "The longest part of the lab was probably researching signal sampling to understand the concept, as I have not yet covered this subject in any of my classes. It was interesting to see how different sampling rates affected the output, especially those that were too low to accurately represent the original signal." The first sentence of this quote is a very promising statement. If the part that was the most cumbersome and took the longest was learning about the topic that the lab covers, then that is a success. The student, while performing a Take Home Lab, did extra research to understand sampling in greater depth than already discussed in the lab handout. This is a promising outlook for the Take Home Labs.

### 4.4.5 Observations of Lab Reports and Student Interactions

This section considers assessments in addition to the Piazza surveys: 1) a comparison of old lab reports (Fall of 2015) versus new reports (Fall of 2017), 2) a comparison of Piazza survey answers and lab reports (to check the honesty of the Piazza surveys), and 3) other observations from interactions with the students.

#### Old Reports Versus New Reports

In 2015 a noticeable percentage of students complained about the length of this experiment in their lab reports. To reduce the time students took, we removed the external mode investigation, started the experimental section with the highest sampling frequency, and simplified the plotting function. In the 2017 reports there were no complaints about the length of this experiment. These changes appeared successful.

As earlier stated, one of the main messages in the Take Home Labs is the difference between theory, simulation, and experiment. In 2015, less than half of the students (10 out of 29) even included hand sketches of their sampled sine waves. This means that over half of the students failed to do a third of the lab (theory). In the 2017 reports 31 out of 34 students included their hand sketches and discussions of all three sections of the lab. This indicates that the changes made in the handouts to clarify the importance of all three sections of the lab (theory, simulation, and experiment) were successful.

There were two main issues that seemed to have students confused in 2015: the non-sinusoidal shapes of the drifting motor position plot and the initial low sampling rate velocity plot. By removing the position plot and starting with the highest sampling frequency for the velocity plots, we seem to have significantly reduced the confusion, as evidenced by lack of comments on these subjects in the 2017 reports.

**Survey Questions Versus Lab Reports**

Although only 13% of the students did not agree that they understood the difference between sampling frequency and the frequency of the sine wave, in their lab reports 1/2 of the students (16 out of 34) made mistakes that demonstrated some problems in this area. Some changed the frequency of the sine wave when they were supposed to change the sampling rate. Some changed both rates together. Others made sketches in which the time axis did not match either the sampling frequency or the sine wave frequency. It was surprising that Juniors in ECE would not be more familiar with these topics. We may need to revise this experiment to emphasize the differences between the sampling frequency and sine wave frequency.

After the lab, the majority of the students claimed that they felt comfortable using the Nyquist Theorem to choose a sampling rate. Most students in their reports did know how to choose a sampling rate based on the Nyquist Theorem. However, when students were asked in the lab handout to describe any drawbacks to choosing a fast sampling rate, almost no students explained the fact that your sampling rate can be limited by the speed of the computer. Even though this isn't a crucial concept for the students to learn, it demonstrates the difficulty of looking past theoretical limits to think about practical considerations.

In the piazza surveys, the vast majority of students said that they were comfortable converting frequencies from rad/s to Hz and back, as well as finding the period of a sine wave from the frequency. However, in the lab reports, several students referred to frequency in seconds or period in Hz. Also, some students referred to the sampling frequency in terms of rad/s. rather than Hz, which indicates a lack of understanding of the the process.

### Other Observations

One step in the lab had students simulate a sampled 1 Hz sine wave at a sampling interval of 0.9 seconds. They were then asked if the original signal could be reconstructed. The sampled sine wave now looks like a 0.11 Hz sine wave (period of 9 seconds). The signal has been aliased, meaning that the higher frequencies get mapped into the lower frequencies due to the slow sampling interval. We did not expect the students would be very familiar with aliasing. We did expect that they would realize the sine wave was sampled too slow. Without even looking at the sampled sine wave, it is easy to tell that the sine wave was not sampled faster than the Nyquist Rate, yet students seemed to miss that fact, and many said that the original signal could be reconstructed. We may need to provide additional guidance on this step of the experiment.

## 4.5 Conclusion

In the 2015 version of the labs, the Sampling and Data Acquisition experiment was a turning point for some students. The difficulty they had getting the serial plot function to work with the byte adjust function, the confusion caused by having the first sampling rate be very slow (so that the sine wave was not easily recognizable) and the length of the experiment caused by investigating both normal mode and external mode, caused some students to become discouraged with the labs and negatively affected their attitudes in later experiments. The changes we made for the 2017 version of the sampling experiment seemed to significantly change student attitudes. Students were asked on Piazza what parts of the lab took the longest time. The most common answer was "drawing the hand sketches." One student mentioned researching signal sampling to understand the concept. These are the types of things that we want students to spend time on. We do not want them to be frustrated with technical details. In 2015, it did not seem to be evident to the students that the whole point in this experiment was to be able to compare

theory, simulation, and experiment, since many students did not even include theory. However, because of the minor changes, in 2017 more comprehensive comparisons between all three sections of the lab were more prevalent in student reports.

# CHAPTER 5

# OPEN LOOP STEP RESPONSE

## 5.1 Objective

The Open Loop Step Response experiment is a revision based on Sean Hendrix's thesis [2]. This experiment is designed for students to derive a first order model for a DC motor system by experimentally collecting data and matching a simulation to the experiment. Given the experimentally derived transfer function, the students can find other motor parameters.

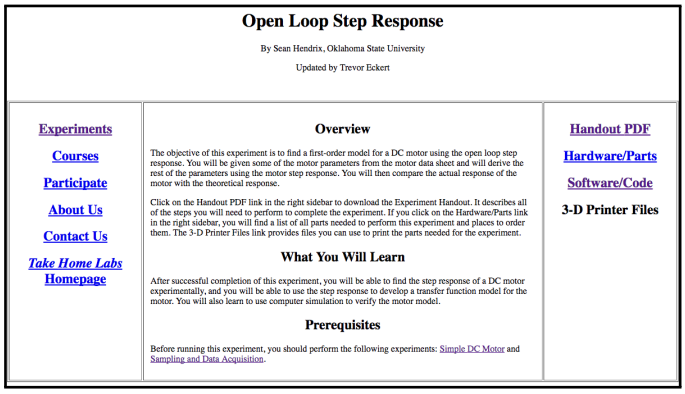



Figure 5.1: Open Loop Step Response Webpage

## 5.2 Experiment Flow

This section in the thesis will go over the basic flow of the experiment, but will not go into great detail for each step. It will include software setup, deriving the motor transfer function, theoretical open loop step response, experimental open loop step response, and simulation of the open loop step response. Students will then make connections between theory, simulation and experiment. The experiment handout can be found in

Appendix G or refer to the project handout on the Take Home Labs website for more specifics. Figure 5.1 shows the Open Loop Step Response webpage.

### 5.2.1 Software Setup

In this section of the lab, students are instructed to download a MATLAB function that was written for some of the labs in the Take Home Labs Repository. This function helps students line up experimental plots to help with making comparisons between simulation and experiment.

### 5.2.2 Deriving the Motor Transfer Function

In this exercise students are asked to derive the open loop motor transfer function $\frac{\Omega(s)}{E(s)}$, given the equations of motion for a general DC motor model, where $\Omega$ is the velocity of the motor and $E$ is the input voltage to the motor.

### 5.2.3 Theoretical Open Loop Step Response

Next, the students are asked to take their derived transfer function and put it into the form shown in Equation 5.1, where $K_m$ is the open loop DC gain, and $\tau_m$ is the time constant.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_m}{\tau_m s + 1} \tag{5.1}$$

This form is a general form described for all first order Linear Time-Invariant systems.

The students will then apply a unit step input of $A$ volts to the motor and derive the motor velocity $\omega(t)$ and plot the response versus time. Students are asked to describe how $K_m$ and $\tau_m$ affect the step response and how the pole location changes the time

response.

### 5.2.4 Experimental Open Loop Step Response

In this section of the lab handout students set up Simulink software that applies a 3 volt step input into the motor. They collect and plot motor velocity in order to derive the open loop transfer function of their motor. Using theoretical knowledge, students should be able to obtain $K_m$ and $\tau_m$ of their motor. Using the values the students have found for $K_m$ and $\tau_m$ they are asked to find the remaining motor parameters $\beta$ and $J$, which are the viscous friction coefficient and the inertia of the armature and the load, respectively.

### 5.2.5 Compare Simulation and Experimental Results

In order to confirm proper derivations of $K_m$ and $\tau_m$, students will simulate the transfer function found in the experimental exercise and compare the two plots to verify their open loop transfer function. Once the comparisons are made, the students can make any adjustments to their values to obtain their final open loop transfer function.

## 5.3 Changes Made to the 2015 Experiment

This section discusses the changes made to the Open Loop Step Response lab, and the reasoning behind each change. The changes were made based on experiences during the System Dynamics class in the Fall of 2015.

### 5.3.1 DC Motor Parameters

With the change to a new DC motor, certain certain parameters that were used in the Open Loop Step Response lab needed to be updated. These parameters are the back emf constant ($K_b$), the torque constant ($K_t$), and the armature resistance ($R_a$).

As stated in [18], $K_b$ and $K_t$ are equal. Therefore only one needs to be derived. To find $K_b$, we apply a constant voltage to the motor with no load. The value of $K_b$ is computed by taking the voltage applied and dividing it by the steady state velocity in rad/s. Using the average of three trials for three different voltages, $K_b$ came out to be 0.0058 $\frac{V}{rad/s}$. Therefore $K_t$ is 0.0058 $\frac{Nm}{A}$.

The armature resistance, $R_a$, was measured for one motor with a multimeter and came out to be 2.6 Ω. This value was used in the revised lab handout.

When deriving $K_b$, a problem with nonlinear affects was discovered. Due to the static friction of the motor, a large enough voltage had to be applied so that the motor could move. However, the bigger issue is that, because the motor shield can only produce a maximum of 2 Amps, there is a limit to how much voltage can be applied. Figure 5.2 shows steady state velocities at different voltages. Table 5.1 shows the values computed for $K_b$ from Figure 5.2. Somewhere between 4 and 6 volts is where there is a significant change in comparison to 3 and 4 volts. The max voltage at the point where the current saturates is

$$V = IR$$

$$V = 2A \times 2.6\Omega$$

$$V = 5.2\ Volts$$

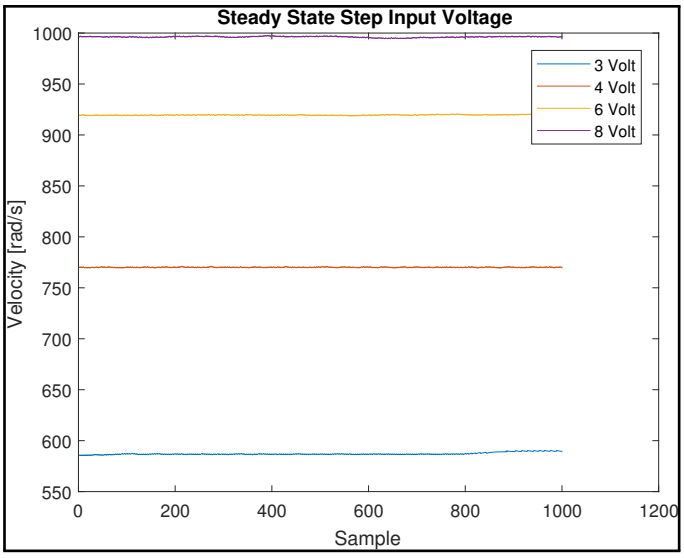



Figure 5.2: Steady State Velocities For Different Voltages

Table 5.1: Back EMF Constant values

| Voltage | $K_b \left[\frac{V}{rad \times s^{-1}}\right]$ |
|---|---|
| 3 | 0.0050 |
| 4 | 0.0052 |
| 6 | 0.0065 |
| 8 | 0.0080 |
| avg | 0.0062 |

### 5.3.2 Adding Fins to the Loads

In 2015, students were assigned different numbers of pennies to put into their loads. This makes it possible for students to have different inertia for their loads and ultimately changes their transfer functions. In order to create more variety, Fins were integrated into the labs. These fins were designed to slip onto the load. Figures 5.3 and 5.4 show top and side views of the load with one fin modeled in software. Figure 5.5 shows the overall motor setup with 4 fins.

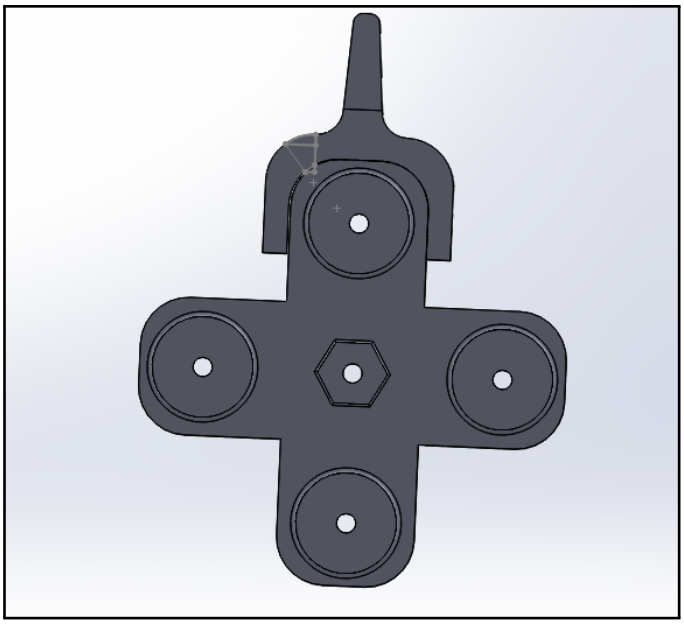


Figure 5.3: Top View of Fin on Load

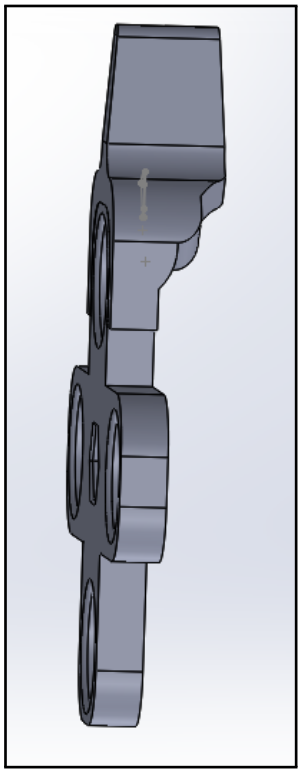


Figure 5.4: Side View of Fin on Load

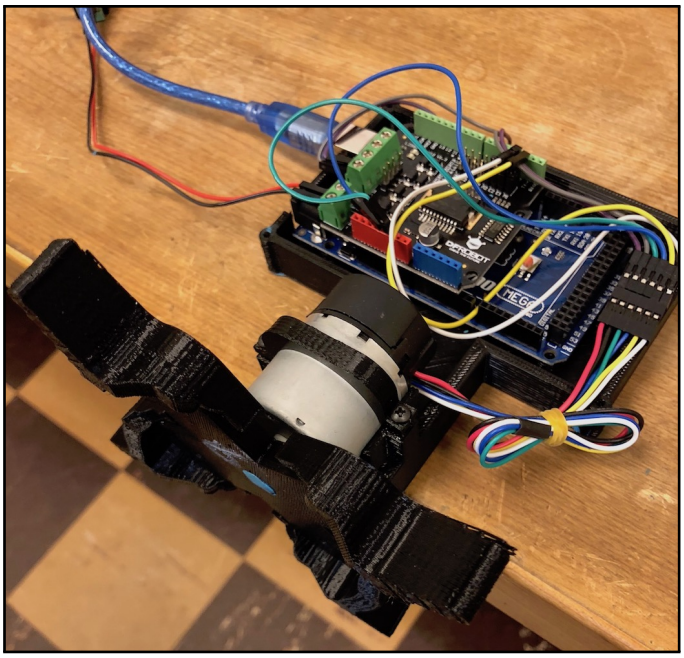


Figure 5.5: Overall System With Fins

The fins will increased the damping affect making the steady state velocity slower. Figure 5.6 shows the affect the fins have on the step response where both cases have the same number of pennies in the load. The fins reduce the steady state velocity by a factor of over three times.

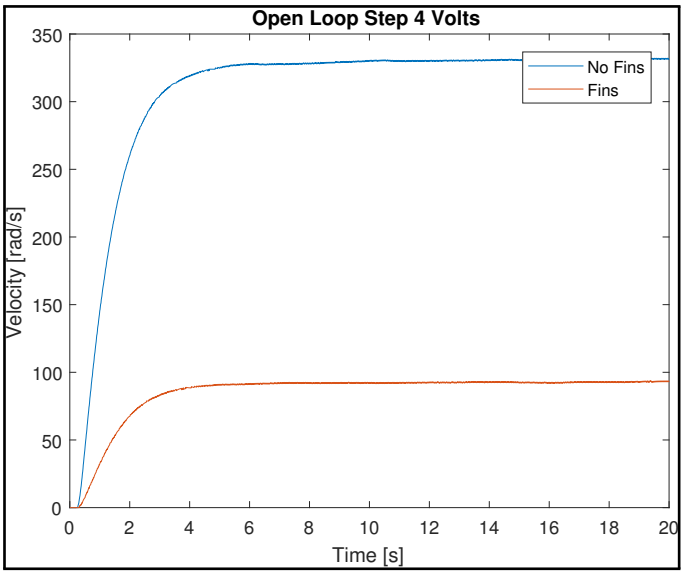



Figure 5.6: Open Loop Step Response With and Without Fins

### 5.3.3 MATLAB Function to Align Plots

In the 2015 lab, when collecting data, students had to stop the plot function at the correct time to get their step response plots to line up (starting at exactly zero). The 2015 reports showed that this was actually difficult to do. Some students had their plots starting at value that was not zero or had a significant amount of leading zeros. This made it difficult for students to compute the time constant of their transfer function. In order to fix this, a MATLAB function (FindShift2.m) that takes a sequence of step responses and produces a step response that starts perfectly at zero was integrated into the experimental section of the lab.

In order to incorporate this new function into the experiment, the students used a pulse generator block to generate the voltage to the motor (Figure 5.7 shows the data collected by the students). Then the students take the data and run the FindShift2 function to locate and plot a single step response, as shown in Figure 5.8. Please refer to Appendix B for the code.

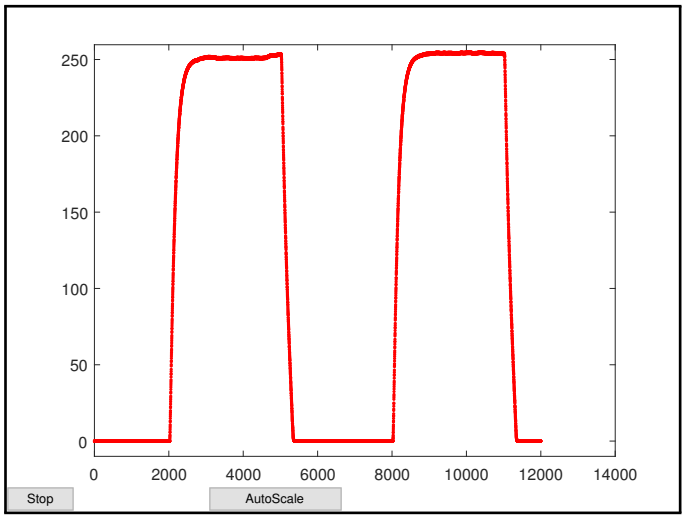



Figure 5.7: Plot of Open Loop Experimental Pre-Processed Data

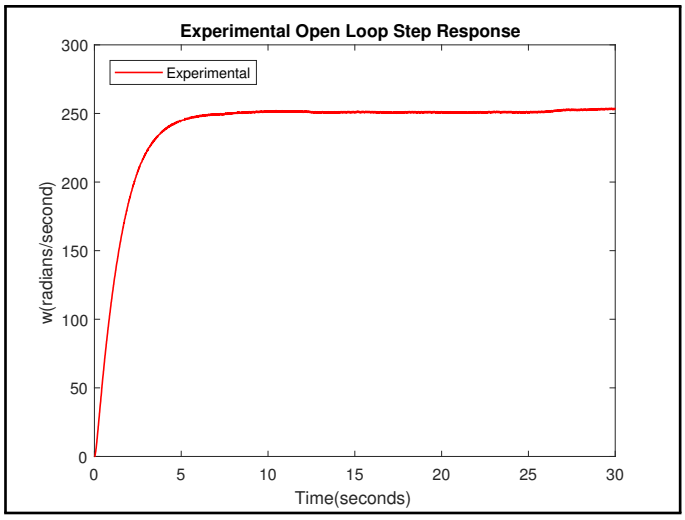



Figure 5.8: Plot of Open Loop Experimental Post-Processed Data

### 5.3.4 Block Diagram Transfer Function

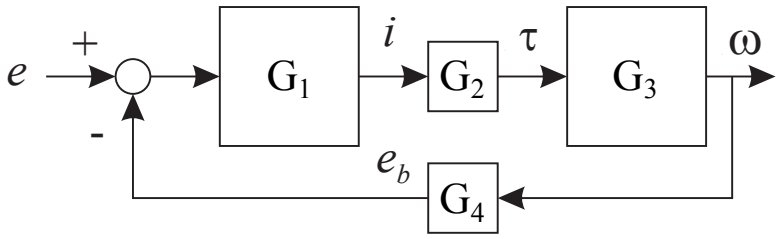



Figure 5.9: Empty Block Diagram for Motor

In the theory part of the lab, students are asked to find the transfer function from voltage $e$ to velocity $\omega$. In 2015, students had to reduce the block diagram in Figure 5.9 to find the overall transfer function. They were not given the specific formula. In 2017, the handout now explicitly states

$$G(s) = \frac{\Omega(s)}{E(s)} = \frac{G_1(s)G_2(s)G_3(s)}{1 + G_1(s)G_2(s)G_3(s)G_4(s)}$$

This change was because the topic is sometimes not covered in the lecture until after the lab starts. Although it is covered in the textbook, and is always covered in the lecture before the lab is due, some students had difficulty reading ahead. To reduce this concern, the specific formula was provided.

### 5.3.5 Importance of Pole Location

The pole location is a very important concept in linear system dynamics. It can tell you a lot of what you need to know about your system. Since the motor is a first order system, there is only one pole. That makes this experiment a good place for students to start analyzing the effects of pole locations. In 2015, there were no questions in the Open Loop Step Response lab handout about the pole locations and how they affect the time response. The 2017 handout now asks about poles. For example: "Find and plot the pole of your open loop transfer function. As the pole moves to the left in the complex plane, how would the system time response change?"

## 5.4 Assessment and Results

This section of the thesis will go over the Piazza surveys that were posted before and after the experiment, identify what topics students feel they have improved on, and tie it all together through the student reports from 2017 as well as reports from the fall of 2015.

### 5.4.1 Pre-Lab Questions

Before the students performed the Open Loop Step Response lab experiment, four questions were posed to get a feel for students general prior knowledge when it comes to deriving mathematical models, first order step responses, making comparisons (between theory, simulation, and experiment), and transfer functions. The questions were asked in the from of a Likert scale as follows:

1. I am familiar with the DC motor mathematical model.
    (a) Strongly Disagree
    (b) Disagree
    (c) Neither Agree Nor Disagree

(d) Agree

(e) Strongly Agree

2. I am comfortable plotting a first order step response of a system.

3. I understand the differences between theory, simulation, and experiment.

4. Given a plot of the step response of a first order system, I could derive the transfer function.

The results of these questions are shown in Table 5.2.

Table 5.2: Pre-Lab Open Loop Step Response Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I am familiar with the DC motor mathematical model. | 7% | 20% | 27% | 40% | 7% |
| I am comfortable plotting a first order step response of a system. | 0% | 14% | 24% | 52% | 10% |
| I understand the difference between theory, simulation, and experiment. | 0% | 0% | 7% | 57% | 37% |
| Given a plot of the step response of a first order system, I could derive the transfer function. | 3% | 20% | 17% | 40% | 20% |

Table 5.2 shows that 47% of students believe that they are familiar with the DC motor mathematical model. This statistic is surprising because modeling of electromechanical systems was never discussed in full detail in the classroom and therefore should be a new topic for the students.

62% of students felt that they were comfortable plotting a step response of a first order system. Although this topic is covered in detail in class before the beginning of the

lab, some students may still have difficulties. It will be made more apparent when we assess the lab reports.

As stated multiple times thus far, the overall objective of the Take Home Labs is for the students to investigate the differences among theory, simulation, and experiment. The more we can get that point across to the students, the more fluid the labs become. 94% of students feel they understand the difference, but this could be due to the students being overconfident.

In Table 5.2, Questions 2 and 4 are essentially the reverse of one another. Either the students noticed that, or the statistic holds true. Since 62% agreed for Question 2 and 60% agreed for Question 4, the majority of students feel they are comfortable with first order systems.

### 5.4.2 Post-Lab Questions

After students completed the lab, additional questions were posed. This allowed us to make before and after comparisons. The questions were based on a Likert scale and are as follows.

1. I have a better understanding of the DC motor mathematical model after completing this experiment.
    (a) Strongly Disagree
    (b) Disagree
    (c) Neither Agree Nor Disagree
    (d) Agree
    (e) Strongly Agree

2. From the plot of a first order step response, I am comfortable finding the DC gain and the time constant.

3. I was able to match my simulation plot with my experimental plot.

   (a) Not at all

   (b) Minimally

   (c) Marginally

   (d) Reasonably

   (e) Perfectly

4. The most important concept covered in this experiment is...

   (a) How to compute the time constant of a first order system

   (b) How to set the pulse width in a Simulink pulse generator block

   (c) How to use the Simulink scope block

   (d) How to plot a step response of a first order system

   (e) How to download a Simulink model to the Arduino

The results of these questions are shown in Table 5.3.

Table 5.3: Post-Lab Open Loop Step Response Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I have a better understanding of the DC motor mathematical model after completing this experiment. | 0% | 0% | 18% | 61% | 21% |
| From the plot of a first order step response, I am comfortable finding the DC gain and the time constant. | 0% | 0% | 19% | 56% | 26% |
| I was able to match my simulation plot with my experimental plot. | 0% | 4% | 18% | 61% | 18% |
| The most important concept covered in this experiment is... | 36% | 4% | 4% | 57% | 0% |

Table 5.3 shows that after performing the experiment 82% of students now have a better understanding of the DC motor mathematical model. This is an impressive number as well as the 0% of students who disagreed with the statement. This goes to show the lab taught the students something, or at least they felt that it did.

Since deriving the transfer function of a first order system is the main idea behind the Open Loop Step Response lab, it only seemed appropriate to gauge the students comfort in the whole experience. While 82% said they felt comfortable, a proper assessment will be made in the examination of the reports.

Table 5.3, Question 3 gauges how well the students understand that theory and simulation will only get you about 95% of the way to an explanation of experimental results. The goal was to see how many students could match their simulation reasonably to their

experiment. If a student says perfectly, then the student does not understand the limitations of simulations.

Question 4 in Table 5.3 is not a Likert scale question, but we thought it was necessary to see how many students realize the goal of the lab. The majority did indeed answer the question as we hope they would at 57%. Answer (a) only asks about the time constant instead of the overall transfer function, like in (d).

### 5.4.3 Comparison of Pre-Lab and Post-Lab Answers

In this section, complimentary pre-lab and post-lab questions will be compared to see how well the students progressed through the lab.

#### First Order Systems

Before the experiment, 62% of students felt comfortable plotting a first order step response of a system and 60% of students felt like they could derive the transfer function given a plot of a first order step response. Since these are just the reverse process of each other, we will say 60% of students feel comfort with first order systems in general. After the experiment, 82% of students felt comfortable finding the DC gain and time constant of the first order step response. This increase from pre-lab to post-lab is significant. This is a topic that can be assessed in greater detail when reviewing the student reports.

#### Theory and Simulations Versus Experiment

94% of students said they understood the difference between theory, simulation, and experiment. However only 61% of students were able to match simulation and experiment reasonably. Even if you lump in the "marginally" answer that still only gives 79%. It could be that the experiments or simulations did not produce the correct results, or it

could be that the students did not understand how to make the comparison. In 2015 a fair amount of students claimed their simulation and experiment matched, even though they were not even close. Comparisons between 2015 and 2017 reports will be discussed later on. Also, this topic will be assessed in greater detail in the Survey Questions Versus Lab Reports section.

### 5.4.4 Free Response Questions

The same free response question as in the previous chapter was asked after the lab was completed: "Describe which parts of the lab took the longest time and/or were cumbersome. Describe which parts of the lab were the most interesting."

Most students wrote that the theory part of the lab took the longest and was most cumbersome. The fact that they felt the theory was the hardest part of the lab means that there didn't appear to be any unexpected technical issues. In fact, there were no students who complained about the length of the lab or difficulties with the software.

The part that students found the most interesting was getting theory, simulation and experiment to match. Students appeared to like seeing a real application of something they have learned in class. "The hardest/ most cumbersome part of this lab was all of the theoretical calculations. Once I got a handle on that, the lab went smoothly. The most interesting part of the lab was comparing the experimental and simulated plots. I was excited to see that my two plots were almost exactly the same. It was a cool way to check that my calculations were correct." This quote summarizes how the majority of students felt about this lab.

### 5.4.5 Observations of Lab Reports and Student Interactions

This section considers assessments in addition to the Piazza surveys: 1) a comparison of old lab reports (Fall of 2015) versus new reports (Fall of 2017), 2) a comparison of Piazza survey answers and lab reports (to check the honesty of the Piazza surveys), and 3) other observations from interactions with the students.

#### Old Reports Versus New Reports

In 2015, students had to stop their plots at the right point in order to get their step response plots to start exactly at zero. This caused issues with calculating the DC gain and time constant of their experimental data. In fact, 15 out of 29 students had miscalculated $K_m$ and/or $\tau_m$ due to plots not starting at zero (either starting after the response started rising or before, which produced leading zeros). In 2017, the FindShift2.m aligned their plots, which allowed them to perform calculations and compare simulations much more easily. This appeared to work much better. Only 3 out of 34 reports had some form of miscalculation, and this would not be due to bad experimental data. Instead, it appeared to be due to students not understanding the theory.

In 2015, by the time this portion of the report was due, only 19 out of 29 students submitted the Open Loop Step Response lab (66%). In 2017, 31 out of 34 students submitted on time (91%). This is a significant increase in student participation. This could be because the previous lab (Sampling and Data Acquisition) did not have as many technical difficulties as the 2015 semester and therefore did not burn the students out on the Take Home Lab experience.

#### Survey Questions Versus Lab Reports

The main goal of this lab was to derive a first order transfer function based on experimental data. In Table 5.3, Question 2, 82% of students said that they were confident

finding the transfer function of their motor. In the student reports, only 3 out of 34 students were not able to find the transfer function correctly (91%). More students were able to find the transfer function than were confident in being able to do it. However, 18% of students said they neither agreed nor disagreed. This could be because some students did not absorb the content of the lab and rather just followed it blindly to get sufficient answers in their reports.

Table 5.3, Question 3 asks the students how well they matched their simulation to their experiment. It is not clear why 61% chose reasonably because in the reports 85% of the students were able to reasonably match the simulation plot to their experiments, while none had a perfect match. This could be due to the vagueness of the options in the questions, but it was often discussed in class that simulation will never exactly match the experiment due to noise, disturbances, and nonlinearities. One minor fix that could be done to prevent students from thinking that they can match plots perfectly is to create an error plot. The error plot would be a subtraction of experiment from simulation. If the error is zero, then they perfectly match, which will never happen.

**Other Observations**

In Piazza, questions can be posted by students, and professors and other classmates can respond. In this lab, a few students were having errors in the findShift2 function and could not output a proper step response. The issue was that they not collect data for long enough. In the lab handout, students are prompted to set the serial plot to 12,000 points and to let the plot fill entirely so that they have 12,000 points. The students who had this issue did not let the serial plot fill to the required data size. Although the instructions in the handout were very specific, and most students had no trouble with the software, we may want to further highlight these instructions in the next version of the handout.

Students appear to have a hard time with the concept of pole locations and how they affect the time response. The more exposure they get to this concept, the more it will sink in. In the next lab (Closed Loop Step Response), pole locations will play an even bigger role. It is hoped that putting the pole location questions in this lab will prime the students for the following one. In the student reports, 8 out of 34 completely skipped the lab handout question on pole locations and 3 additional students seemed to not have enough knowledge about pole locations. That leaves only 23 out of 34 students demonstrating a basic understanding of that topic (about 2/3). This topic will be investigated more in the next chapter.

Another concept students seem to struggle with is the idea of what a transfer function is. In fact, 11 out of 34 individual students and/or groups came in to get help during office hours on the theoretical part of solving at least one of the transfer functions $G_1(s) - G_4(s)$. Since this whole class revolves around transfer functions, it is not entirely clear why students had such a hard time with this concept. It is discussed during approximately 40% of the class lectures and is heavily emphasized in the lectures leading up to this lab.

## 5.5 Conclusion

The Open Loop Step Response lab is truly the first lab where students in the System Dynamics course apply theory learned in class to a physical system. It is important not to distract them from the main goal of this lab. In 2015, too many students couldn't perform proper calculations on their experimental data because of the imprecision of stopping the serial plot at the correct time. The 2017 reports proved that having a function to align the plots makes it easier for the students to make the correct calculations. This lab also demonstrated the utility of Piazza. In addition to its usefulness in post-

ing student surveys for assessment, students began to use it more for asking questions of instructors and other students. Because we were notified whenever a question was posted, we could monitor student progress in real time. If difficulties come up, we could address them quickly, and in such a way that the entire class could follow the discussion. In 2015, students could only ask questions during class, office hours, and by email. The first two methods only take place at specific times, and an email response has limited visibility. Piazza posts are available for the entire class to see, and they remain easily accessible throughout the semester.

# CHAPTER 6

# CLOSED LOOP STEP RESPONSE

## 6.1 Objective

The Closed Loop Step Response experiment is a revision based on Sean Hendrix's thesis [2]. The objective of this experiment is to investigate the affects of the step response when an integrator and feedback are added to the system derived in the previous experiment (Open Loop Step Response). The students will gain understanding of how feedback alters the system poles and how they relate to the time response. Theory, simulation, and experiment will be compared to discover benefits and limitations of linear modeling.

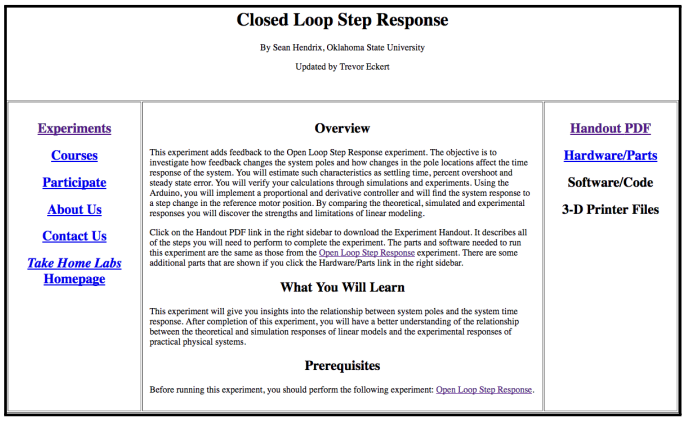



Figure 6.1: Closed Loop Step Response Webpage

## 6.2 Experiment Flow

This section will go over the basic flow of the experiment, but will not go into great detail for each step. It will include how to set up some extra hardware for a small exercise, closed loop transfer function theory, closed loop transfer function simulation,

and closed loop transfer function experiment. Please refer to the project handout in Appendix H. Figure 6.1 shows the Closed Loop Step Response webpage.

### 6.2.1 Hardware Setup

Some extra hardware is needed to perform this lab, which includes three female to male wires and a potentiometer. One outer wire is connected to 5V and the other wire is connected to ground. The middle wire is connected to an analog input pin in the Arduino. This hardware is used in between simulation and experiment sections of the lab to motivate the need for an automatic control system.

### 6.2.2 Theory

In the theoretical section, students are asked to find the closed loop motor transfer function using block diagram reduction (from reference position to output position). Then, using the open loop transfer function found in the previous lab, they derive closed loop pole locations for two different sets of feedback gains. Based on the pole locations, the students are asked to estimate the settling time, percent overshoot, and frequency of oscillation as well as hand sketch the step response.

### 6.2.3 Simulation

After the theoretical calculations have been completed, the students then set up the Simulink file to simulate the step responses for the same two sets of feedback gains. Comparison between theory and simulation is crucial at this point in the lab, to verify student understanding of second order systems.

### 6.2.4 Experiment

There are two parts to the experimental section in this lab. First, the students act as the control system (human in the loop). Using the potentiometer, which adjusts the

motor voltage, they will try to turn the motor 90 degrees as fast as possible, and stop with as little error as possible. The students will come to the conclusion that this task is not as easy as it sounds. This part of the experimental section of the lab is motivation as to why we would want to use an automatic control system.

Once the Human in the Loop section is completed, setting up the software for the experiment is similar to the Open Loop Step Response experiment. Only a few changes are necessary. The students add feedback and gain terms to the Simulink diagram. Now the students run the experiment for both sets of gains and make comparisons between theory, simulation, and experiment.

## 6.3 Changes Made to the 2015 Experiment

This section discusses the change made to the Closed Loop Step Response Lab, and the reasoning behind each change. The changes were made based on experiences during the System Dynamics class in the Fall of 2015.

### 6.3.1 Human in the Loop

The human in the loop section of the lab was added to help students understand how much faster a computer can sense, process data, and control versus a human. A potentiometer is wired to an analog to digital pin in the Arduino. Based on the voltage on the analog pin, a scaled voltage is applied to the motor. The students can now watch the load spin and try to turn the potentiometer accordingly. The following is the relevant section of the lab handout.

---

**Exercise 3: Motivation - Human in the Loop**

In this experiment you will try to get the computer to turn the motor exactly 90 degrees and stop. This is harder than you think. To demonstrate how hard this is you will try to do it manually.

**Required Hardware**

- Potentiometer
- 3 Male-to-Female wires

1. Take the potentiometer labeled as pins 1, 2, 3.
2. Then take the potentiometer and insert the pins labeled 1 → GND, 2 → Analog In 4, and 3 → 5V. Refer to Figures 6.2 and 6.3. **Note:** DO NOT WIRE THE MIDDLE PIN TO POWER OR GROUND.

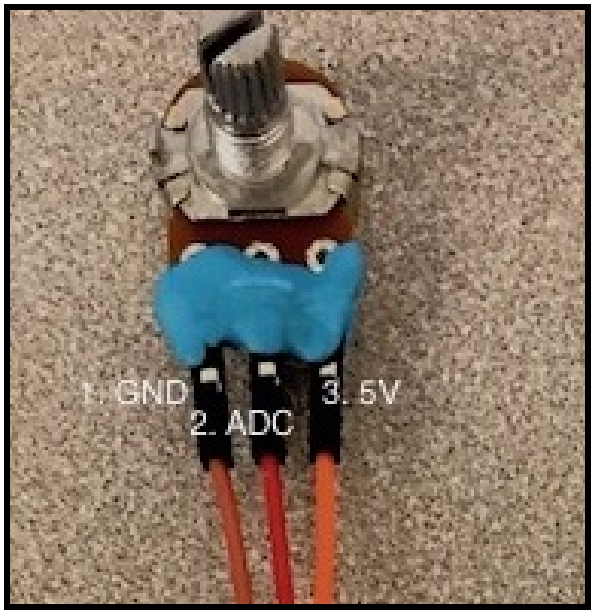



Figure 6.2: Potentiometer

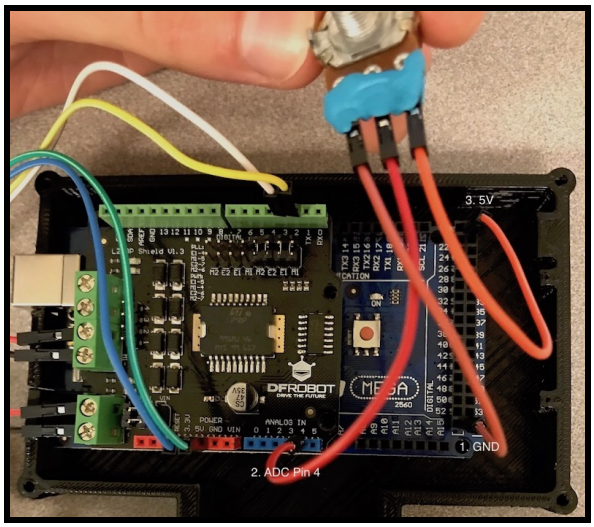



Figure 6.3: Potentiometer Connected to Arduino

**Software Setup**

3. Open the Simulink file created in the *Open Loop Step Response* experiment named **OL_Step_Resp_Arduino.slx**.

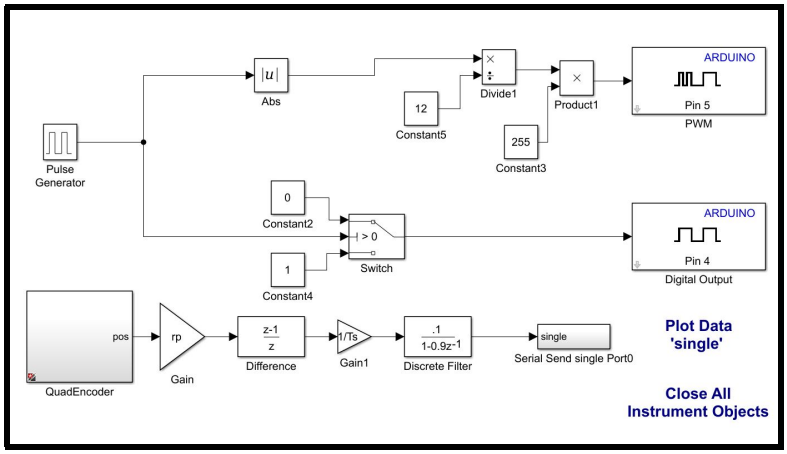



Figure 6.4: Simulink Model from Open Loop Step Response Experiment

4. Add 3 **Constant** blocks, a **Sum** block, a **Divide**, a **Product**, and an **Analog Input** block.

5. Make the **Analog Input** block pin 4 and the sample time *Ts*, then connect the output of the **Analog Input** block to the input *x* of the **Divide** block.

6. Next, take one of the **Constant** blocks set it to *1023* and sample time to **Ts**. Then connect it to the ÷ port of the **Divide** block.

7. Take another **Constant** block, set it to *5* and sample time to *Ts*, then connect the **Constant** block to one input of the **Product** block and the output of the **Divide** block to the other input of the **Product** block.

8. Then take the last **Constant** block, set it equal to *-2.5* and connect its output to the one of the inputs of the **Sum** block. Then connect the output of the **Product** block and connect it to the other input of the **Sum** block.

9. Lastly, take the output of the **Sum** block and connect it to both the input of the **Abs** block and the middle input of the **Switch** block. Refer to Figure 6.5 for the final Human in the Loop configuration.

10. Plug in the Arduino and upload the model to the board.

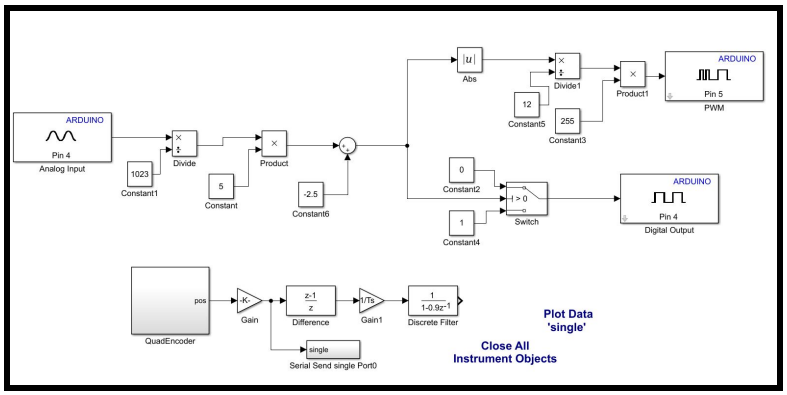



Figure 6.5: Simulink Model For Human in the Loop

### Experimental Human in the Loop

11. Plug in power to the motor shield. **Note**: Make sure nothing is in the way of the load spinning.

12. Try adjusting the voltage applied to the motor by turning the potentiometer.

13. Once you get the feel for turning the potentiometer, try and get the motor to turn exactly 90 degrees and stop.

14. How difficult is it to get the motor to turn the load and stop at exactly 90 degrees?

---

After attempting to control the motor precisely by hand, the students should realize how hard this task actually is, and the experience will hopefully catch their interest into automatic controls.

### 6.3.2 MATLAB Function to Align Plots

The findShift2 function described in the previous chapter was also added to the Closed Loop Step Response experiment. Please refer to the previous chapter (Open Loop Step Response) for a full discussion.

## 6.4 Assessment and Results

This section will go over the Piazza surveys that were posted before and after the experiment, identify what topics students feel they have improved on, and tie it all together through student reports from 2017 as well as reports from the Fall of 2015.

### 6.4.1 Pre-Lab Questions

Before the students preformed the Closed Loop Step Response lab, four questions were posed to get a feel for the students prior knowledge when it comes to understanding of pole locations, block diagram reduction, hand sketching second order step responses, and the differences between theory, simulation, and experiment. The questions were asked in the form of a Likert scale as follows:

1. I understand how poles determine the system response.

(a) Strongly Disagree

(b) Disagree

(c) Neither Agree Nor Disagree

(d) Agree

(e) Strongly Agree

2. I am comfortable simplifying a block diagram.

3. I am comfortable hand sketching a second order step response given a transfer function.

4. Simulation should always match the experimental results.

The results of these questions are shown in Table 6.1.

Table 6.1: Pre-Lab Closed Loop Step Response Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I understand how poles determine the system response. | 3% | 3% | 23% | 61% | 3% |
| I am comfortable simplifying a block diagram. | 3% | 13% | 13% | 39% | 32% |
| I am comfortable hand sketching a second order step response given a transfer function. | 3% | 32% | 19% | 29% | 16% |
| Simulation should always match the experimental results. | 16% | 39% | 26% | 13% | 6% |

System poles are an important concept, and, by the time this lab was started, students should already have an understanding of how poles determine the system response. With only 64% seeming to have some form of understanding before the lab,

hopefully by the time the students complete this lab an even greater majority understands this.

Block diagram reduction is another concept covered in this lab. This is necessary to acquire the closed loop transfer function and then get the closed loop poles. 71% of students said they have some comfort reducing block diagrams. This is something that can be easily investigated in the student reports.

In the theoretical section of the lab, students are asked to hand sketch a second order step response based on their transfer functions/pole locations. The results for this pre-lab question (no particular majority) are a little concerning, since by this time they have already had several homework problems and have been quizzed over this concept. The student reports and post-lab answers will be a gauge as to how well this lab helps with the overall idea of second order responses.

By now the students should have realized that the comparison between theory, simulation, and experiment is the most important idea behind the Take Home Labs. While simulation should never exactly match the experimental results, 19% felt that it should. We hope that this question will prime students to think more about this concept as they work on this lab.

### 6.4.2 Post-Lab Questions

After students completed the lab, additional questions were posed. This allowed us to make before and after comparisons. The questions were based on a Likert scale and are as follows:

1. I understand how feedback can move the system poles.

    (a) Strongly Disagree

(b) Disagree

(c) Neither Agree Nor Disagree

(d) Agree

(e) Strongly Agree

2. I feel comfortable simplifying a feedback loop.

3. Given a step response of a second order system, I feel comfortable finding the transfer function of the system.

4. I understand why my experimental results did not completely match my simulation.

5. The most important concept covered in this experiment is...

(a) understanding how the findShift2 function works.

(b) finding the pole locations of a closed loop transfer function and how they affect the step response.

(c) how to redeploy adjusted gain values to the Arduino.

(d) finding the closed loop transfer function using block diagram reduction.

(e) how to compute the initial voltage being applied to the motor.

The results of these questions are shown in Table 6.2.

Table 6.2: Post-Lab Closed Loop Step Response Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I understand how feedback can move the system poles. | 0% | 7% | 14% | 66% | 14% |
| I feel comfortable simplifying a feedback loop. | 0% | 0% | 10% | 52% | 38% |
| Given a step response of a second order system, I feel comfortable finding the transfer function of the system. | 0% | 7% | 13% | 57% | 23% |
| I understand why my experimental results did not completely match my simulation. | 0% | 3% | 27% | 47% | 23% |
| The most important concept covered in this experiment is... | 0% | 76% | 7% | 17% | 0% |

After the lab has been completed students seem to have a better understanding of feedback and pole locations (80%).

Feedback loops are a very common thing in control systems and therefore simplifying a feedback loop is also common. 90% of students said they felt comfortable simplifying a feedback loop.

80% of students said if they were given a second order step response, that they could find the transfer function. In the lab, the students will find the step response from the transfer function, but they will also analyze the experimental step response to verify that it has the correct shape. It will be interesting to see which they find easier.

70% of students said they had some confidence as to why their experimental results

did not exactly match their simulations. This topic will be discussed in further detail when we analyze their reports.

While 76% of students said they thought the most important concept in this lab is finding the pole locations of a closed loop transfer function and how they affect the step response, it could be understood why 17% students would think finding the closed loop transfer function using block diagram reduction could also be important. Block diagram reduction is important, but the overall idea of pole locations,transfer functions,and step responses are more important to understand.

### 6.4.3 Comparison of Pre-Lab and Post-Lab Answers

Before the lab was performed, only 64% of students were confident that they understood how pole locations affect the time response. However, after the lab was performed, 80% of students had this kind of confidence. This is a slight increase in confidence. We will check the lab reports to see if their understanding matched their confidence.

Pre-lab, 71% of students were confident about simplifying block diagrams. Post-Lab, 90% of students were confident about simplifying a feedback loop. The increase appears to be promising. However, it is hard to tell if this is overconfidence. The student reports will help gauge how many students performed the block diagram reduction correctly.

An interesting statistic is shown in regards to matching simulation and experiment. In the pre-lab only 54% indicated that simulation will never exactly match the experiment. However, after the lab was performed, 70% of students stated that they knew why they did not have an exact match. Only 54% understood that there would be an inexact match, but then 70% understood why there was an inexact match. Student reports will help understand how many students truly understand the difference between simula-

tion and experiment.

### 6.4.4 Free Response Questions

The same free response question as in the previous chapter was asked after the lab was completed: "Describe which parts of the lab took the longest and/or were cumbersome. Describe which parts of the lab were the most interesting."

A common occurrence on these free responses is that most students said that the theory and calculations at the beginning took them the longest. This has been a pattern for all previous labs in 2017 - students are saying the part that took them the longest is that part that should take the longest. This is in contrast to 2015, when students complained about spending extra time because of technical glitches.

Some students made comments in their free responses about trying to get the motor to turn 90 degrees manually. "The most interesting part of the lab was definitely when it had us attempt to control the motor "manually" using the potentiometer." The point of including that small section in the lab handout was to get students to understand why we want to use automatic feedback control. It appears that this exercise was effective.

One student made a comment about how they did not get their experiment to match their simulation. In reading the lab reports, we found that most students had experimental results that did match the theory, within the limitations of a linear model. However, a number of students expected a perfect match. For those students who did not get a reasonable match, it was usually because the transfer function for the motor that they found in the Open Loop Step Response experiment was not correct.

### 6.4.5 Observations of Lab Reports and Student Interactions

This section considers assessments in addition to the Piazza surveys: 1) a comparison of old lab reports (Fall of 2015) versus new reports (Fall of 2017), 2) a comparison of Piazza survey answers and lab reports (to check the honesty of the Piazza surveys), and 3) other observations from interactions with the students.

#### Old Reports Versus New Reports

Comparing this experiment in 2015 versus 2017, there were so many differences it is almost too hard to make comparisons. In 2015, by the time of this experiment, the students had already completed 2 more labs compared to 2017. One of these extra labs was Open Loop Frequency Response, which took longer than most experiments and included more difficult concepts. The improvements that we see in 2017 suggest that in 2015, there were too many labs, and thus going from 7 total labs to 5 could have made a significant impact on the attitudes of the students. In fact, only 11 out of 29 students (40%) turned in the Closed Loop Step Response lab on time in 2015, while 28 out of 34 students (82%) turned in the lab on time in 2017.

The quality of the reports overall are far better in 2017 than in 2015. In 2015, the majority of students left off important theoretical calculations and hand sketches, ignored questions in the handout, and made very elementary comparisons among the three aspects of the lab (theory, simulation, experiment). The 2017 reports were not perfect by any means, but they show a significant improvement.

Making slight changes to each experiment seems to have produced large improvements in the Take Home Lab experience. It started by making the Sampling and Data Acquisition Lab shorter, clearer, and simplifying the real-time plotting function. Another change was the addition of the table of discussions and questions, which reminds

students to go back over their work. The combination of these changes could be the reason for the increased quality of reports.

**Survey Questions Versus Lab Reports**

The first thing students do in the Closed Loop Step Response lab is block diagram reduction to acquire the closed loop transfer function. After the lab was completed, 90% of students said they were comfortable simplifying a feedback loop. In fact, 88% of students demonstrated this properly in their reports. Although this is not the most important topic of the lab, the consistency between lab results and the post-lab survey answers is promising.

While 85% of students demonstrated how to compute step response percent overshoot, frequency of oscillation, and settling time based on pole locations (in their reports), only 56% could then hand sketch the second order step response based on those calculations. There appears to be some sort of gap between the two concepts, even though that is about all of the information you need to roughly sketch the second order time response, because 80% of students said they felt comfortable deriving the transfer function given the second order step response. This could be because it may be easier to go from the sketch to the transfer function, but when students have to picture what the response will look like, they freeze up.

In the lab reports, 68% of students got their simulation and experiment to match reasonably, but only 26% of students could properly justify why simulation did not match experiment. Most student discussions and comparisons were not very expansive on this topic. The static and the coulomb friction in the motor were the reasons why linear simulation will never perfectly match experiment. These concepts were covered in the lecture, but the majority of the students were not able to make the connection from the

abstract concept to the physical system. One interesting thing to note on this topic: 70% of students said they understood why they didn't match, but only 26% of students gave reasons why in their reports.

**Other Observations**

In answer to the free response question, one student said "The experiment didn't take me long time. All the software parts have been set up previously. The most confusing thing is that the experimental plot does not consist with the simulation. I recalculate the equation many times to make sure every coefficients are correct but still got the problem." Figures 6.6 and 6.7 show the simulation and experimental plots that this student put in their report. In fact, these plots are very close to the best that can be done without modeling static and coulomb friction. The purpose of the Take Home Labs is to help students learn the relationships among theory, simulation, and experiment. Theory can get 90% of the way to prediction the experimental response, but unmodeled effects (like nonlinear static and coulomb forces in this experiment) limit the accuracy. This experiment is a good test of whether or not the THL concept can help students grasp that concept. For some students it clearly did, but for many the result is not so clear. It is not easy to understand the precise limitations of a linear model, and these students are only in their first year of professional school. It seems unlikely that one course can fully clarify these ideas for all students, but it can begin the process.

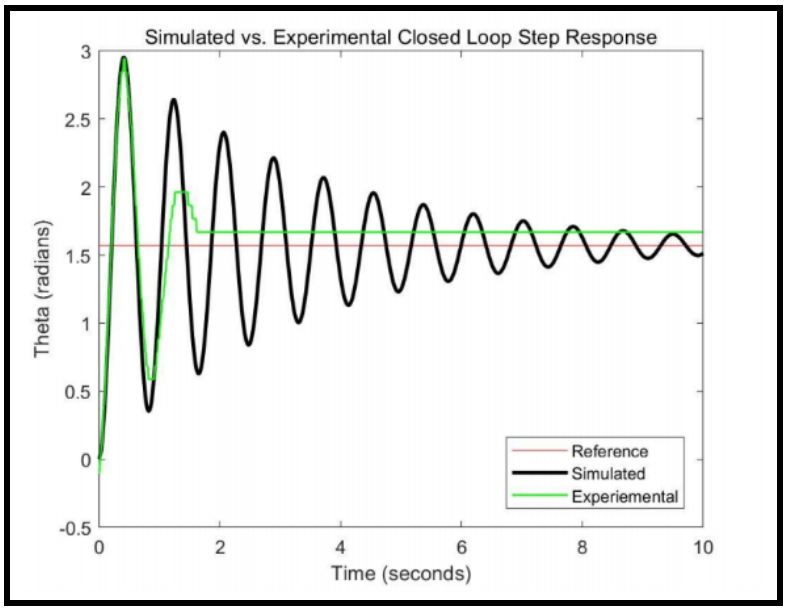



Figure 6.6: First Case Student Plot

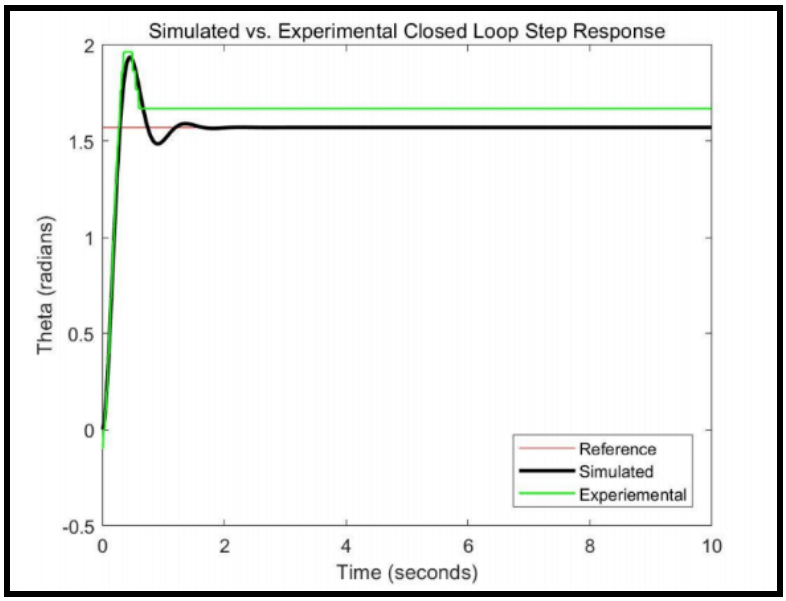



Figure 6.7: Second Case Student Plot

In the student reports from 2017, just a little under half of the students either had bad theoretical step response sketches or none at all, but when comparing theory and simulation, the students would often say they did match. It appears that students have trouble comparing functions of time. This is apparently related to the fact that most students could find percent overshoot, frequency of oscillation, and settling time from a

sketch, but had difficulty making a sketch, if they were given these values. In the future, we will look for exercises that could build these skills.

## 6.5 Conclusion

The most difficult parts of this experiment were making the theoretical second order step response and comparing experimental results with the simulations. Since the Open Loop Step Response lab involves only a first order system, it is easier to match experiment with simulation. The second order response is more complex, and students had more trouble making accurate sketches. In addition, due to the static and coulomb friction in the motor, it is practically impossible not to have steady state error, which students had a difficult time explaining. The students will see this again in the next experiment, which should reinforce the concept. Positively speaking, a significantly larger percentage of students turned in, and performed the experiment well in 2017 in comparison to 2015. All of the small issues in 2015 seemed to have accumulated to discourage some students as time went on, but, when slight changes were made in 2017, there were fewer complaints, more participation, and deeper discussions in the lab reports.

# CHAPTER 7

# ROOT LOCUS CONTROL DESIGN

## 7.1 Objective

The objective of this experiment is to design a feedback control system for a motor positioning system. Students will determine the best closed loop poles of their motor, using their transfer functions found in the Open Loop Step Response experiment, via the root locus diagram. Two different control architectures will be implemented: proportional control and proportional derivative control. Students will simulate and then test the controllers experimentally. This lab is to teach students about design iteration when practical issues arise.

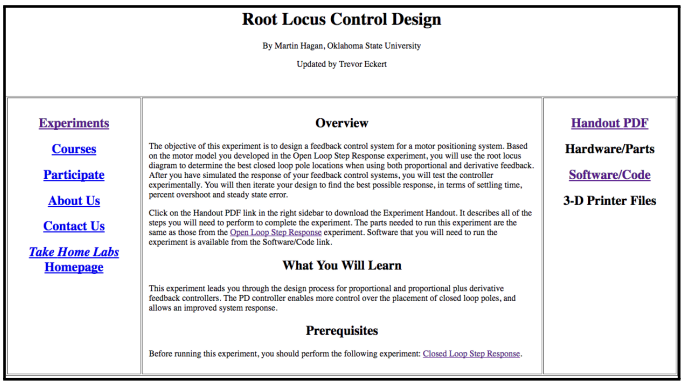



Figure 7.1: Root Locus Control Design Webpage

## 7.2 Experiment Flow

This section will go over the basic flow of the experiment, but will not go into great detail for each step. It will include control design, verification via MATLAB, simulation, and experiment. Please refer to the project handout in Appendix I. Figure 7.1 shows the Root Locus Control Design webpage.

### 7.2.1 Theory

Students perform block diagram manipulation (much like in the Closed Loop Step Response experiment) to find the transfer functions $G(s)$ and $H(s)$ for the equivalent block diagram in Figure 7.2.

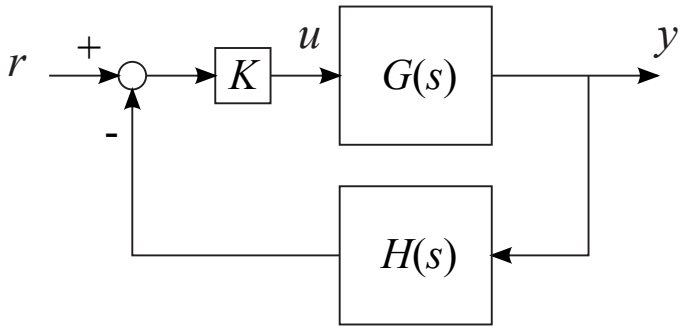



Figure 7.2: Standard Feedback Control Block Diagram

Then the students calculate closed loop pole locations for various values of K to get the general shape of the root locus diagram. They also calculate percent overshoot, time of the peak, and settling time (5%) to relate pole locations to the step response. A question is posed as to what would be the "best" closed loop pole locations and for what value of K. Although there is no single correct answer to this question, we would expect the students to pick locations where the real part and the imaginary part are equal (damping ratio is .707), which would produce the fastest response time with minimum oscillation.

### 7.2.2 Simulation

In MATLAB there is an application called the Control System Designer that is useful for analyzing single-input, single-output (SISO) controllers for feedback systems. Students will use this application to verify their results and calculations were correct. In Simulink, a simulation file is made as well, much like the Closed Loop Step Response lab.

### 7.2.3 Experiment

Once the students have simulated their motor position control system, then they will run the experiment to make comparisons among theory, simulation, and experiment. Sometimes the students first choice of K will not work physically (at least for the proportional controller) because K will be so small that the resulting voltage would not turn the motor. It should be obvious that the student now needs to redesign their control system. After the students pick a reasonable design and make comparisons among theory, simulation, and experiment, then the process is repeated for the proportional derivative controller.

## 7.3 Changes Made to the 2015 Experiment

This section discusses the changes made to the Root Locus Control Design lab, and the reasoning behind each change. The changes were made based on experiences during the System Dynamics class in the Fall of 2015.

### 7.3.1 Closed Loop Pole Plots

One method for designing a control system is the root locus diagram. The root locus diagram shows how the pole locations move when changing a parameter (for this lab the proportional gain, $K$). In 2015, this appeared to be a hard concept for students to grasp. In order to help students understand what the diagram means, a table was added to the theoretical section of the lab. This table has four values for K and the students are asked to calculate and plot the closed loop poles at these values of K. Explicitly plotting the poles at various values of K is supposed to help students understand what changing K does to the system. The students are also asked to calculate the percent overshoot, time of the peak, and settling time (5%) to relate these characteristics to the corresponding pole locations.

### 7.3.2 Encoder Resolution

In 2015, the encoder that was built into the motor had 1023 counts per revolution. The motor that was selected to replace this motor only has 64 counts per revolution, as discussed in the Simple DC Motor chapter. Although the decrease in encoder resolution does not distract from the main purposes of all the labs, it can be made more apparent to the students that encoder resolution could be a hardware limitation. We decided to add a question to the Root Locus Control Design handout that gets the students thinking about encoder resolution. "What is the resolution of the encoder in radians, if there are 64 counts per revolution? How big is the steady state error for your experimental results? Can you make a connection between the encoder resolution, which is used to measure the motor angle, and your steady state error?" Upon performing the experiment, occasionally the steady state error would match the encoder resolution ($\frac{2\pi}{64} \approx 0.098$ radians). This question, while subtle, should help students think about of physical hardware limitations and add to comparisons among theory, simulation, and experiment.

## 7.4 Assessment and Results

This section will go over the Piazza surveys that were posted before and after the experiment, identify what topics students feel they have improved on, and tie it all together through student reports from 2017 as well as reports from the Fall of 2015.

### 7.4.1 Pre-Lab Questions

Before the students preformed the Closed Loop Step Response lab, four questions were posed to get a feel for the students$'$ prior knowledge when it comes to understanding what a root locus diagram is, deriving a closed loop transfer function given a block diagram, understanding the affect the pole locations have, and system linearity. The questions were asked in the form of a Likert scale as follows:

1. I understand what a root locus diagram is.

   (a) Strongly Disagree

   (b) Disagree

   (c) Neither Agree Nor Disagree

   (d) Agree

   (e) Strongly Agree

2. Given a block diagram, I am comfortable deriving the closed loop transfer function.

3. I understand how the pole locations affect the settling time, percent overshoot, and frequency of oscillation.

4. I know how to tell if a system is linear.

Table 7.1 shows the results to the Pre-Lab Root Locus Control Design questions.

Table 7.1: Pre-Lab Root Locus Control Desgin Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I understand what a root locus diagram is. | 0% | 3% | 10% | 68% | 19% |
| Given a block diagram, I am comfortable deriving the closed loop transfer function. | 0% | 0% | 3% | 61% | 35% |
| I understand how the pole locations affect the settling time, percent overshoot, and frequency of oscillation. | 0% | 3% | 13% | 68% | 16% |
| I know how to tell if a system is linear. | 3% | 16% | 23% | 52% | 6% |

83% of students stated that they understood what a root locus diagram is before starting this lab. This questions appears to be simple, but even just the basic idea of

what a root locus diagram is seems to be something students struggle with. However, the students have had homework, a quiz, and had a root locus question on their mid term before starting this experiment.

By now a very large majority of students should be comfortable getting the closed loop transfer function given a block a diagram. The Closed Loop Step Response experiment should have helped with that. Indeed, 96% of students felt comfort in this area.

Pole locations are an extremely important concept in dynamic systems. This has been emphasized throughout the course, so students should be able to make the corresponding calculations needed from pole locations. In fact 84% of students said they understood how pole locations affect the settling time, percent overshoot, and frequency of oscillation.

The question on linear systems was included, in part, to prime the students to be thinking about the fact that the transfer function models that they are using, and the associated concepts like poles, apply only to linear systems. They will be asked, as part of the experiment, to explain why simulated results do not match the experiment. The main causes are static and coulomb friction, which are nonlinear. It is surprising that 58% of the students said that they knew how to tell if a system is linear, since it is not a simple concept, and was covered only at the beginning of the semester. This is a concept that is best understood by working with a real physical system, and is one of the motivations behind the THL concept. When we review the lab reports, we can assess the effectiveness.

### 7.4.2 Post-Lab Questions

After students completed the lab, additional questions were posed. This allowed us to make before and after comparisons. The questions were based on a Likert scale and are as follows:

1. I understand how velocity feedback affects the shape of the root locus.
   (a) Strongly Disagree
   (b) Disagree
   (c) Neither Agree Nor Disagree
   (d) Agree
   (e) Strongly Agree
2. I feel comfortable using the root locus diagram to design a control system
3. There is always one clearly best design
4. The closed loop motor transfer function depends on (choose all that apply)
   (a) Initial Conditions
   (b) The Inertia of the Load
   (c) The Input to the System
   (d) The Feedback Gains
   (e) The Version of your MATLAB
5. The most important concept in this experiment is
   (a) Understanding how feedback affects the pole locations
   (b) How to enter a transfer function in MATLAB
   (c) How to open the control system designer

(d) Understanding how pole locations affect the time response

(e) Starting your motor at exactly 0 degrees

6. It is easier to move the motor quickly and precisely 90 degrees by manually adjusting the motor voltage than by automatically adjusting the voltage through feedback control.

Table 7.2 shows the results to the Post-Lab Root Locus Control Design questions.

Table 7.2: Post-Lab Root Locus Control Desgin Results

| Question | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|
| I understand how velocity feedback affects the shape of the root locus. | 9% | 5% | 9% | 55% | 23% |
| I feel comfortable using the root locus diagram to design a control system | 5% | 9% | 27% | 36% | 23% |
| There is always one clearly best design | 18% | 64% | 9% | 5% | 5% |
| The closed loop motor transfer function depends on (choose all that apply) | 43% | 62% | 29% | 100% | 0% |
| The most important concept in this experiment is | 55% | 0% | 0% | 45% | 0% |
| It is easier to move the motor quickly and precisely 90 degrees by manually adjusting the motor voltage than by automatically adjusting the voltage through feedback control. | 50% | 32% | 14% | 5% | 0% |

78% of students said they understood how velocity feedback affects the shape of the root locus after the lab, while 83% of students said they understood what a root locus diagram was before the lab. We will need to assess the lab reports to gauge their true understanding.

The whole point of this lab is to design a control system using the root locus diagram. 59% claimed they felt comfortable doing this. We hoped that these numbers would have been a little higher, but it is the majority.

Engineering can be tough because there are usually multiple ways to solve a problem. This is especially difficult for students to adjust to, due to the math intensive courses where problems usually have one right answer. Undergraduate courses tend to not focus so much on design. In the pre-lab question, 82% of students at least disagreed with the statement "There is always one clearly best design." Most of the students understand the ambiguity of design.

Throughout the semester the idea of a transfer function was principle focus of the course. A transfer function only depends on the system itself. There is no need for knowledge about the input or initial conditions. This was said at least a dozen times during the semester, and it was covered in many homework problems, quizzes and exams, in addition to several labs. The students were asked to choose all that apply to the question: "The closed loop motor transfer function depends on... " While 100% of the students who responded did say the feedback gains, only 62% said the inertia of the load. On top of that, 43% said initial conditions and 29% the input to the system. We are not sure why so many students checked the wrong answers when this has been discussed more than any single topic throughout the semester. We need to investigate how we can reinforce the concept in other labs.

The most important concepts covered in the lab according to the students were understanding how feedback affects the pole locations (55%) and understanding how pole locations affect the time response (45%). These both are good answers, and the fact that

nobody selected any of the other options is a very good sign.

82% of students said that adjusting the voltage by hand and getting the motor to turn exactly 90 degrees and stop was harder than using an automatic control system. Having to try to manually control the motor gave the students more respect for the automatic controller. The human in the loop exercise in the Closed Loop Step Response lab did help with this.

### 7.4.3 Comparison of Pre-Lab and Post-Lab Answers

While 87% of students said they understood what a root locus diagram was before preforming the experiment and 84% said they understood how pole locations affect the settling time, percent overshoot, and frequency of oscillation, 59% said they felt comfortable using the root locus diagram to design a control system. Understanding what a root locus diagram is and understanding how pole locations affect the step response are the two pieces needed to design a control system. Even though roughly 85% of students said they knew both pre-lab questions does not mean that they know how to combine the two successfully, but 59% is not a bad number. The Root Locus Control Design lab is just an introduction to control systems and common practices, especially for a System Dynamics class.

### 7.4.4 Free Response Questions

The same free response question as in the previous chapter was asked after the lab was completed: "Describe which parts of the lab took the longest and/or were cumbersome. Describe which parts of the lab were the most interesting."

The majority of students, again, thought the theoretical section took the longest. It

is a good sign that they tend to think this way. Theory, especially for the root locus, takes a lot of calculations and therefore it should take the longest.

When students are using the Control System Design Tuner, They have to input the transfer functions, $G(s)$ and $H(s)$, as transfer function objects in MATLAB. For $G(s)$ this is an object that does not change throughout the entire lab, but for $H(s)$ the students have two different transfer functions: 1 and $K_2 s + 1$. In the handout students are told how to explicitly input $G(s)$. However, when it comes time to input $H(s) = K_2 s + 1$, there seemed to have been an issue. Two students in the free response claimed that they did not know how to edit $H(s)$ in the Control System Design Tuner. Even though there were complaints, we did not want to just give that away to the students. We want them to be able to think for themselves. This appears to be an issue in general that students want to be told what to do step-by-step.

While some students said finding the values of $K$ and $K_2$ was very hard, some students found this to be the most interesting part. The design process is something that should take awhile and should not be easy.

### 7.4.5 Observations of Lab Reports and Student Interactions

This section considers assessments in addition to the Piazza surveys: 1) a comparison of old lab reports (Fall of 2015) versus new reports (Fall of 2017), 2) a comparison of Piazza survey answers and lab reports (to check the honesty of the Piazza surveys), and 3) other observations from interactions with the students.

**Old Reports Versus New Reports**

In 2015, over the course of the semester, participation among the students appeared to get worse. When it came to submitting a portion of a lab, the submission rates kept getting worse with each lab. Since the Root Locus Control Design lab is theoretically the hardest as well as being the last lab, the students put in less effort to that section of their lab reports. In fact, 11 out of 29 students either turned in practically nothing or skipped major areas of the lab handout. Reduction of the number of labs and key fixes to technical issues from previous experiments gave students more time and energy to put forth more effort into the Root Locus Control Design lab. In 2017, only 5 out of 34 missed major parts of the root locus lab when it came time to submit the entire lab notebook.

A reoccurring issue with the 2015 lab reports is that the majority of students did not include their hand sketches, which is an important part to all of the theoretical sections. In 2015, 25 out of 29 students left off hand sketches, which includes the root locus diagrams and example step response sketches for very small and very large values of $K$. With emphasis on theory and the comparisons among theory, simulation, and experiment only 6 out of 34 students did not include hand sketches in 2017.

**Survey Questions Versus Lab Reports**

The second portion of the Root Locus Control Design lab adds velocity feedback to the system. While the students start with $K_2 = 0.2$, during the design process they are allowed to adjust this value to see how different feedback gains affect the root locus. This should give the students a better understanding of how velocity feedback affects the shape of the root locus. 78% of students said they understood this in the post-lab questions, but only 10 out of 34 (about 29%) students actually altered $K_2$ (even though it says to experiment with different $K's$ and $K_2's$. This may stem from the fact that students

ran out of time on the last lab.

While only 59% of students said they felt comfortable using the root locus to design a control system, about 70% appeared to have a clear design process in their reports. When students are performing (any) lab experiment, sometimes they execute steps without thinking too much about it, and thus they become under confident. This statistic shows that students performed this experiment successfully but were not confident that they performed it correctly.

**Other Observations**

The static friction in the motor becomes a good teaching moment for students, especially for the Root Locus Control Design lab. For the proportional controller they are asked to find the "best" $K$ value for their motor system. For a second order system, the "best" pole locations are when the real part equals the imaginary part, which will produce the quickest response without significant oscillation. This is something that is taught in the class, but students tend to forget this part. The most common question asked on Piazza was how to find the best K value. The second most common question was what to do when the $K$ value is too small to actually make the motor move for the "best" pole locations. Either students didn't remember that the static friction would stop the motor if the voltage was too small, which they quantified in the first experiment, or that they realized this but did not know where to go from there. Piazza was a great way for students to ask immediate questions if they had any doubt about this. In fact, Figure 7.3 shows an example of a student asking this question. With only a proportional controller, sometimes you cannot make K small enough to give the "best" pole locations, because then the voltage is to small to move the motor. In this case you need to make K large enough to overcome the static friction. However, if K is too large, you may get ex-

cessive oscillations. This teaches the students about practical design tradeoffs and gets the students thinking about why we introduce velocity feedback (proportional derivative controller). Nonlinear effects, like static friction, cannot be incorporated in transfer function models, so students learn about differences between theory and practice.

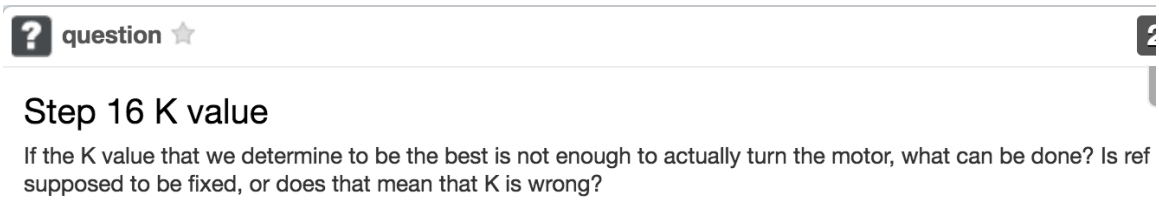



Figure 7.3: Piazza Question About Small K

Upon reading the reports, it can be shown that some students still do not have a grasp on how to make proper comparisons among theory, simulation, and experiment. Some students simulated with one set of gains and ran the experiment with a different set of gains. While this only appeared in a few reports, some students seem to not have a grasp on making proper comparisons. An example of this can be found in Figure 7.4. It was easy to see that this student simulated the proportional controller but experimentally collected data for a proportional derivative controller. These are two different control systems and should never be compared in this type of situation. To correct potential issues like this there should probably be a statement that reminds them to use the same gain values for simulation and experiment.

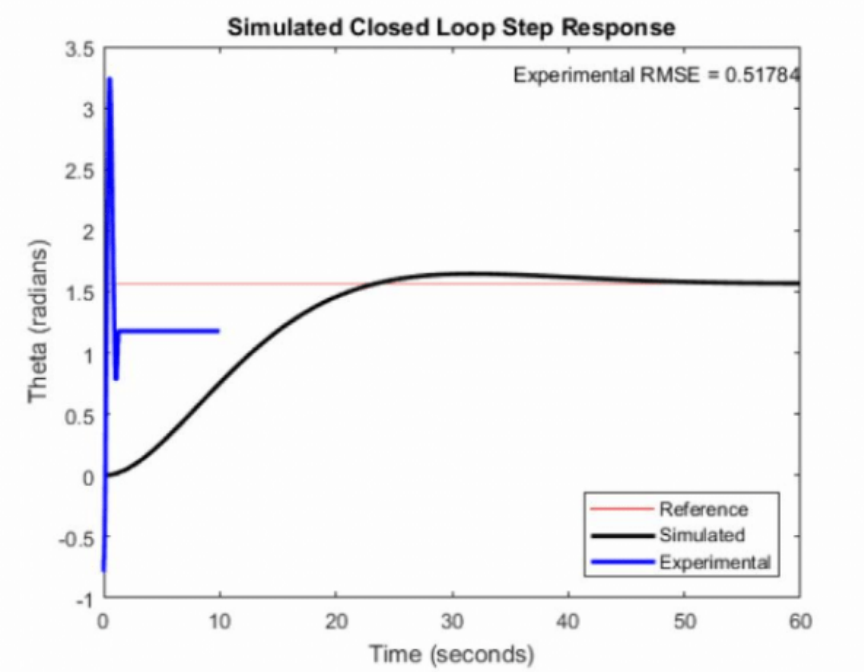



Figure 7.4: Example of Two Different Sets of Gains

## 7.5 Conclusion

The Root Locus Control Design lab may be the hardest experiment out of all of the labs assessed, because it is more open-ended. Going from a step by step process of a modeling a system to designing a control system can be a leap for some students. This is okay, since this is only a system dynamics class and not an automatic control systems class. We do not expect the students to be experts in control design when they complete this course. This course introduces the concept, so that they will be ready for the next course. Also, this type of open-ended experiment, with practical (non-ideal) obstacles, helps prepare the students for the real world.

# CHAPTER 8

# RESULTS

This chapter summarizes the results that were discussed in previous chapters and makes some conclusions about the overall effects of the changes that we made to the dynamic systems labs since the 2015 installment. The chapter begins by describing the issues that came up in 2015 - general issues that affected many labs, and specific issues that affected single labs. This is followed by a summary of the changes that were made - both general and specific. Finally, the chapter concludes by summarizing the assessments of the 2017 labs, and the comparison with 2015.

## 8.1 Issues From 2015

In 2015 there were issues that affected all of the labs, some of the labs, or an individual lab. The next section describes issues that affected multiple labs.

### 8.1.1 Multiple Lab Issues

One of the biggest issues in the 2015 student reports was that a large portion of students skipped steps. This could either be important theoretical calculations, hand sketches, questions, and/or expansive discussions. Without proper theoretical calculations and hand sketches, the experiment will not be successful, because there is not a way to make proper comparisons. If students skip questions and do not make extensive discussions, then the important concepts will not be reinforced.

A major complaint throughout the 2015 semester was that the labs took too long or

that they were too much work. We do not want the Take Home Labs to be a burden. They should be interesting and should help reinforce concepts covered in the class.

One technical issue that made the labs more difficult and lengthy was the issue with the serial plot. As stated in the Sampling and Data Acquisition chapter, the byte adjust button was the problem. It often gave students unusual plots and numbers. A fair amount of students complained how long it took them to get the plots to work. Figure 8.1 shows a plot of what was supposed to look like a sine wave. This student claimed in their report that they tried to get the serial plot work, but could not. This issue appeared to be frustrating for students, and once this issue occurred students seemed to be putting in less effort as the semester went on.

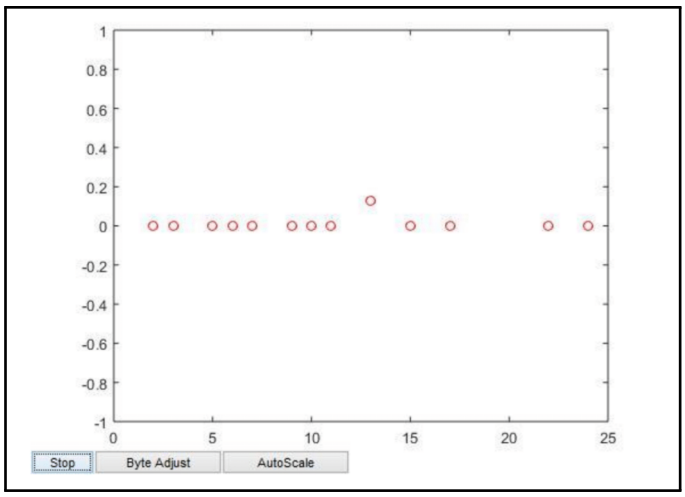



Figure 8.1: Bad Serial Plot

### 8.1.2 Individual Lab Issues

#### Sampling and Data Acquisition

The Sampling and Data Acquisition Lab was the largest hurdle for students to complete. They complained about the length of this lab and the concept of sampling is not easy to grasp. Since System Dynamics is a junior level class, but Digital Signal Processing is a senior level class, this might be the first time sampling is introduced.

The biggest problem that is prevalent in the 2015 lab reports is that students have a hard time understanding the difference between the sampling frequency and the frequency of the sine wave. These two concepts are completely different and should never be confused. Figure 8.2 shows a student's plot, where they changed the frequency of the sine wave instead of the sampling frequency. The instructions told student to maintain a 1 Hz sine wave.

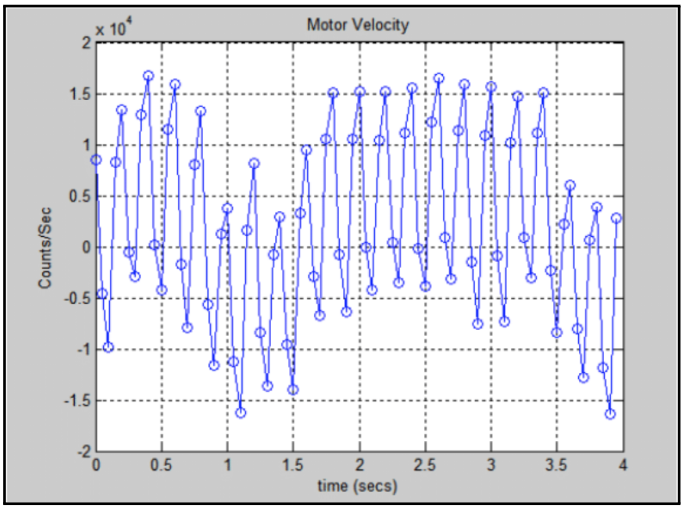



Figure 8.2: Bad Sampling Frequency

**Open Loop Step Response**

The Open Loop Step Response lab is a very important lab and is needed for all the remaining labs. Deriving the transfer function properly is necessary to be successful throughout the THL. As mentioned in the Open Loop Step Response chapter, the students had to stop the serial plot at the correct time to get the plot to start at zero. This caused issues with timing. Either students would stop it too early, leaving leading zeros, or they would stop it too late, missing the onset of the step response. This also makes it hard to properly compare simulation with experiment.

Figure 8.3 shows an example of a student stopping the serial plot early (experimental plot shown along with their simulation). This becomes a problem for students to be able

to make proper calculations of the time constant, $\tau_m$. In order to calculate this properly you have to subtract the starting time (when the response starts rising) from the time when the response reaches 63% of steady state. This is the first issue to get past. Even if $\tau_m$ has been computed properly, it is difficult to make comparisons to the simulation.

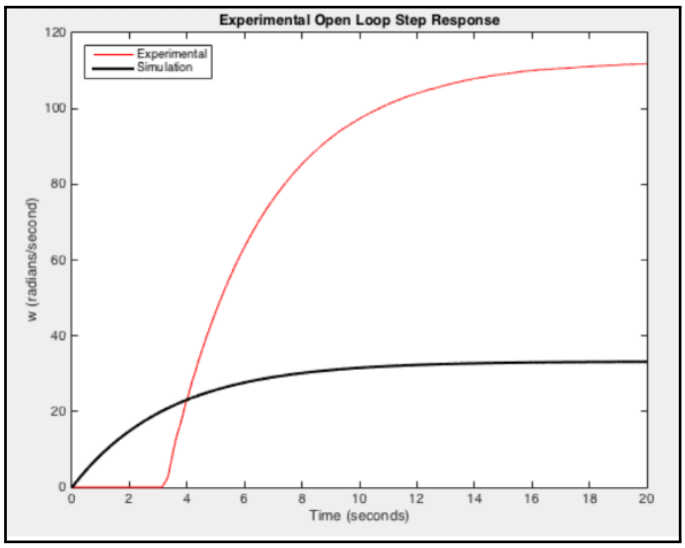



Figure 8.3: Example of Stopping Experiment Early and Simulation

Figure 8.4 shows an example of a student stopping the serial plot late along with their simulation. This is an even bigger problem, as it is impossible to tell the starting point of the response. The time constant is almost impossible to compute. There is not even a way to check your work with simulation as the plots will not line up properly due to the starting point of the experiment.

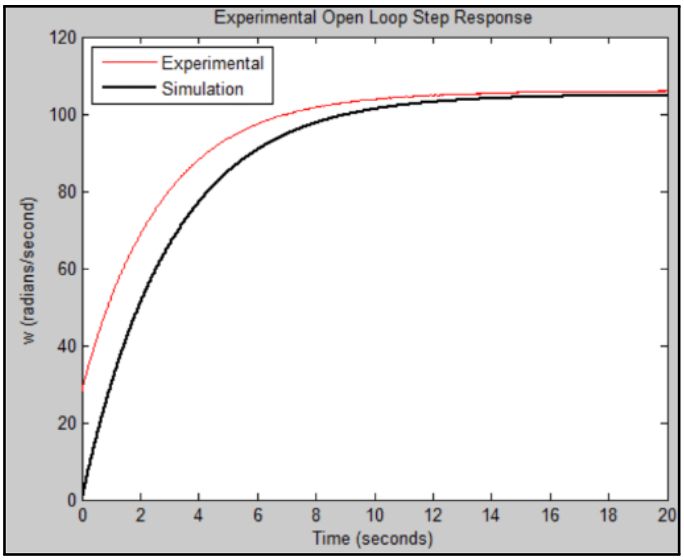



Figure 8.4: Example of Stopping Experiment Late and Simulation

**Closed Loop Step Response**

The Closed Loop Step Response lab introduces the students to feedback control and what the pole locations can say about the step response. Pole locations always seem to be a concept that some students have difficulty grasping. In fact, the pole locations tell you everything you need to know about the transient response. It tells you about the percent overshoot, settling time, time of the peak, and frequency of oscillation of the step response. In 2015, many students had difficulty making these connections.

Block diagrams are usually a new concept to students and are something that can useful if understood. When feedback is added to a block diagram, students tend to not understand what this actually means in terms of the physical system. This is a concept that students struggle with.

#### Root Locus Control Design

The Root Locus Control Design lab requires taking the knowledge learned in previous labs, combined with an understanding of the root locus diagram, to design a control system. This concept is difficult because it combines knowledge of what a root locus diagram is (a plot of pole movements with gain) with the idea of what the pole locations tell you. Students are asked to come up with a control design that gives them the "best" response for the motor and load to turn 90 degrees. This means that the students will choose a proportional gain, $K$, that makes the real and imaginary part of their pole locations equal, with most negative real parts. This will produce the fastest response with minimal oscillation. The lab is an open-ended design problem, so we expect students to struggle, to some extent. In 2015 there were no special issues in this lab.

### 8.2 Changes Made

This section discusses the changes made to Take Home Labs as a whole and changes made to individual labs based on the results of the 2015 class.

#### 8.2.1 Multiple Labs

A major change in 2017 was the use of the Piazza web-based Q&A platform. Introducing Piazza was a good way to get students to interact with each other and the instructor without having to go to office hours. If any student had a question, the question could be answer by a peer or by an instructor. Given that when one student asks a question, many other students may have the same question, using Piazza saved time, since everyone in the class can see the question and answer. Piazza was also used for gauging student involvement, progress, and confidence by posting pre-lab and post-lab surveys.

Since the Take Home Labs was the ECEN 3723 class project in both 2015 and 2017,

there was a project handout on the guidelines. Chapters C and D show the project handouts for 2015 and 2017 respectively. There is quite a bit of change from 2015 to 2017. Because, in 2015, students left out a number of items from their lab reports (e.g., answers to questions posed in the individual experiment handouts), the 2017 project handout had a new section on project notebooks, with a specific list of nine items to include.

Given that students were skipping steps in their reports, at the end of every experiment handout a table of discussions and questions was added. This a reminder for the students to go back and check their work to make sure they have included everything that was requested whether it is a theoretical calculation, hand sketch, etc. Each table has a list of all of the steps that have questions or discussions embedded, along with a summary of the question or discussion.

Given that the majority of students in 2015 thought that the Take Home Labs took too long, there were a few things that were done to fix this. In 2015 the students preformed 7 labs. It was decided in 2017 to reduce the number of labs in the curriculum to 5. The two labs that were removed were Blinking LED and Open Loop Frequency Response. Blinking LED was an introductory lab also, but it was felt that Simple DC Motor was enough of an introduction that Blinking LED was unnecessary. While frequency response is an important subject, the Open Loop Frequency Response lab took a significant amount of time, since it required applying sine waves of multiple frequencies (with very low frequencies requiring long run times). Also, that lab could be removed with little impact on the later labs.

In 2015, the Simple DC Motor and Sampling and Data Acquisition labs experimented with both normal and external mode. Normal mode is a mode that uploads code to the board that cannot be altered during operation. Instead, when something in the code has

been changed, you have to re-upload the code to the board. External mode is interesting because you are capable of updating code in real time. One of the disadvantages of this is that you sacrifice sampling speed. The Sampling and Data Acquisition lab demonstrates this idea well when trying to increase the sampling rate. Although practical issues like this are a good teaching moment, we decided that completely removing external mode from the Take Home Labs would shorten the labs, and thus retain student participation at a higher level. It was believed that external mode distracted the students from the overall concept of the lab.

In the 2015 lab handouts, there were spots that explicitly described what to do in a series of very detailed steps. In 2017, we replaced the detailed series of steps (where a student could miss one or two) with a general description of the procedures, followed by a table, with open entries for the result of each step. This makes the handout shorter, reduces the amount of reading required by the students, does not drain the students concentration, and makes it less likely that students will skip a step.

### 8.2.2 Individual Labs

#### Simple DC Motor

After getting rid of external mode, the Simple DC Motor lab was shorter than the other labs. The Pulse Width Modulation (PWM) section/exercise was added to the handout. In 2015, some students wondered why PWM was used on the motor and how it worked. We incorporated some some things about PWM, along with concepts that are covered in later labs. For example we ask the students to derive a transfer function for an RC circuit (which is similar to the DC motor). Then the students simulate a PWM signal at 50% duty cycle at different frequencies to make connections on why it is possible to regulate a PWM signal to act like an analog signal.

**Sampling and Data Acquisition**

Another addition to fix the length of the Sampling and Data Acquisition lab was to move the entire hardware setup to the Simple DC Motor experiment. This significantly shortened the Sampling and Data Acquisition lab, without an undue burden on the Simple DC Motor Lab, which had already been shortened by removing the external mode sections.

As sampling is a confusing concept, and we do not expect students to understand it in its entirety, there are a couple of things that can be fixed to help the students understand it better. In 2015, students sampled the 1 Hz sine wave starting at the slowest sampling rate. Right from the beginning students are looking at a sampled sine wave that looks very little like a sine wave, and they began to doubt their procedures. Instead, in 2017, the students began with the highest sampling frequency. That way the first thing the students see is the closest thing to a familiar continuous sine wave. This hopefully helps them realize the difference between sampling frequency and frequency of the sine wave, since frequency of the sine wave does not change.

In 2015, students plotted both motor position and motor velocity. Unfortunately, because of a lack of symmetry in motor friction effects, the position plots show a significant drift over time, which caused it to look less like a sine wave. Since the velocity plot did not show this drift, and since a single plot was sufficient to get the concept of sampling across, we removed the position plot from the experiment.

**Open Loop Step Response**

In 2015, students did not get proper alignment of experimental plots, and they therefore obtained poorly calculated values for their transfer functions. As stated in the Open

Loop Step Response chapter, a function called findShift2.m was introduced. Instead of a constant voltage being applied to the motor, a pulse generator is applied instead. This gives students a series of step responses that the findShift2.m function then sorts out into a single step response that starts at zero. Now the students have properly aligned experimental step response plots, which makes it much easier for them to find $K_m$ and $\tau_m$. Also, if those are calculated correctly, their experiment and simulation should match up.

One other small addition to the experiment is a question about the pole location. In 2015, the Open Loop Step Response chapter did not have a question about the pole location. This may seem to be trivial, since it is a first order system, but we wanted students to start getting practice, and we wanted to prime them for the next lab, Closed Loop Step Response.

**Closed Loop Step Response**

Because students have a hard time understanding feedback control and block diagrams, we decided to add in a Human in the Loop section to the Closed Loop Step Response lab. In this section the students act as the controller by adjusting the voltage being applied to the motor to try and get the load to spin exactly 90 degrees and stop. This is supposed to motivate the student and to help them realize what the automatic control system is doing.

**Root Locus Control Design**

A root locus diagram shows you the pole locations as a feedback gain is varied. Since students have a hard time understanding this, a table that asks for pole locations for certain values of $K$ was added to the theoretical section. The table, in addition to pole

calculations, asks for percent overshoot, time of the peak, and settling time. Then the students are asked to plot the pole locations for each specific value of $K$. This hopefully helps students understand the meaning of the root locus diagram, how $K$ changes the pole locations and system response characteristics. The Root Locus Control Design lab is an introduction to designing a control system by the root locus diagram and understanding how pole location affect the step response. The change should reinforce this.

## 8.3 2017 Performance

This section assesses the changes that were made and whether they made an impact on the overall success of the labs, highlights the key improvements, and discusses some ideas for additional lab improvements.

### 8.3.1 Student Involvement and Engagement

Both in 2015 and 2017, students had a chance to turn in their lab reports every two weeks. While the lab reports were not graded until the end, this gave us a chance to see which students were engaged, as well as to critique their reports so they could make adjustments for the next submission. Figure 8.5 shows the percentages of students that turned in reports for each corresponding lab. During the middle of the semester, more students kept up participation in 2017 than in 2015. We believe that a mixture of all of the changes described in the previous sections made it possible for more students to remain motivated throughout the semester.

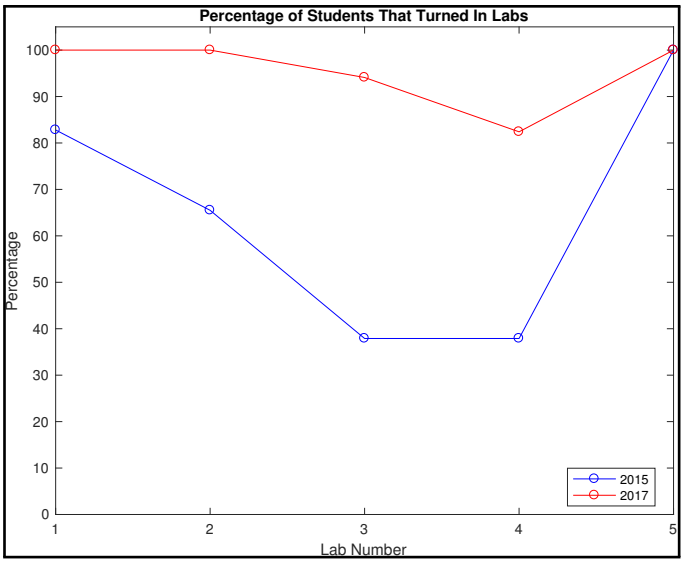



Figure 8.5: Lab Engagement For Each Submission

Figure 8.6 shows the trends during the 2017 semester of unique users per day on Piazza. The graph shows that there are not any long periods of time where the activity on Piazza was significantly low. This is probably due to the fact that the students in 2017 were encouraged to interact with each other by asking questions as well as the surveys before and after each experiment. Even though Piazza was not used in 2015, as shown in Figure 8.5, the unique users per day might have been much lower.

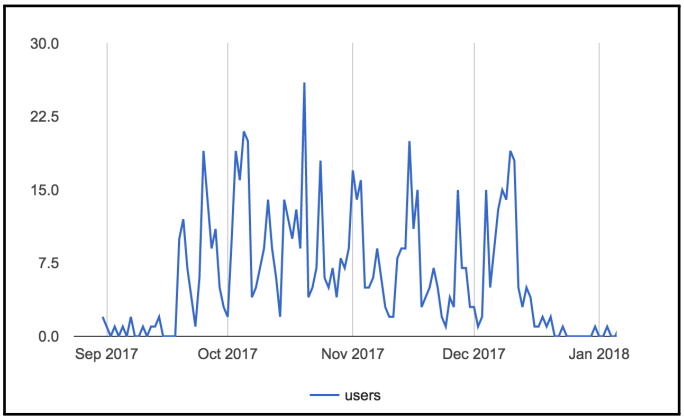



Figure 8.6: Unique Users Per Day On Piazza

As stated before, in 2015, lab 2 (Sampling and Data Acquisition) was a big hurdle for

students. In fact only, 65% of students turned in their report upon completion, and only 38% of students turned in the next two labs. This is a major drop off in student engagement. Part of this could be due to several issues, including: students having issues with the serial plot, some of the labs taking too long, and/or the number of labs. The curve in Figure 8.5 shows that in 2017 the lowest amount of engagement was about 82% for lab 4 (Closed Loop Step Response). It is impossible to say which exact changes made such a significant improvement in 2017 from 2015, but we believe that a combination of the changes impacted the amount of engagement.

### 8.3.2 Key Improvements

Writing seems to be one of the biggest issues in engineering school. It is important to be able to convey your work as clearly and concisely as possible. In fact, in 2015 about 45% of students had poor quality reports. Poor quality reports could be evidence dby missing key calculations, discussions, and figures for any given lab. In 2017, only about 29% had reports that were poor quality. Overall, in 2017 more students included a significant number of figures and were more expansive on their discussions. Again, it is hard to pinpoint which change affected this result. The project handout in 2017 was more detailed on what to include in the reports. This may have helped with overall quality of reports.

Integrating Piazza appeared to be a success. The pre-lab and post-lab questions were helpful for us and for the students. The pre-lab questions primed the students for what concepts are important in the lab they were about to perform. The post-lab questions helped the students reflect on what they had just performed. In addition, the ability to get real-time answers to any questions the students had prevented them from freezing up and failing to complete the lab. With the questions being public, any other student could see the question and the answer, assisting multiple students at the same time.

Overall, including Piazza in the Take Home Lab experience has produced very similar benefits to that of a lab on campus with a TA.

The biggest complaints in 2015 were due to the serial plot and how much extra work the Take Home Labs was. In fact, in 2017 there were not any complaints about either of these. The changes we made appeared to be at least partially responsible for to achieving this. These changes include: fixing the byte adjust on the serial plot, reducing the number of labs from 5 to 7, and reducing the length of some of the labs by taking out external mode and replacing steps with tables.

Integrating the findShift.m function into the Open Loop Step Response appeared helpful upon comparing 2015 and 2017 reports. Fewer students in 2017 (versus 2015) had improper calculations of the DC gain ($K_m$) and the time constant ($\tau_m$). This gave students a better chance to compare their theory, simulation, experiment. This also gave students better transfer functions overall for the Closed Loop Step Response and Root Locus Control Design labs, which, in turn, also gave them better comparisons.

### 8.3.3 Room For Improvement

The idea of a transfer function is an extremely important concept in system dynamics. The students have more than enough practice during the 3723 course to understand this. Almost every homework, quiz, test, and lab integrates this concept. While students seem to have somewhat of a basic understanding, there are still areas that trip students up. One of the final post-lab questions asks the students what the closed loop motor transfer function depends on, which should be the inertia of the load and the feedback gains. While 100% of students said feedback gains, only 62% of the students said the inertia of the load. What is even worse is that 43% of students said the initial conditions

and 29% of students said the input to the system. While these ideas have been discussed multiple times, they do not seem to stick.

Because the Root Locus Control Design lab is only an introduction to automatic controls and this is only a system dynamics course, it is not fully expected for students to understand how to fully design a control system. However, most students appeared to have successfully designed a control system to cause the motor and load to turn 90 degrees, using the root locus diagram. In 2015 the students had to demonstrate this. At the end of the semester students brought in their motor system and competed among other students with the same number of pennies in their load to produce the fastest response. In 2017, this was not done and judging was only based on student reports. While some students had incomplete reports, it was clear that students achieved the objective of the Root Locus Control Design lab.

# CHAPTER 9

## Optimal State Feedback Control (Rotary Flexible Link)

This chapter describes a new lab that was developed as part of this thesis research. It is intended as a more advanced lab that could be used as an honors project for a student in a dynamic systems class, or as a lab in a follow-on control systems class. The remainder of this chapter forms the lab handout, along with results of performing the experimental procedures.

### 9.1 Objective

In this experiment you will build and control a rotary flexible link system. Materials used in certain applications, like robotics and space travel, are not rigid. The rotary flexible link experiment is an example of a non-rigid part. The idea is to try and stabilize the oscillations seen in flexible materials. You will experimentally acquire your motor parameters and design an optimal partial state feedback controller. Then through knowledge of the dynamics of the flexible link, design an optimal full state feedback controller and compare these two controllers to see differences in the link deflections.

### 9.2 Setup Part 1

This section of the lab handout describes all of the required materials needed to make the lab work correctly. There are two subsections, hardware and software. The hardware subsection lists the physical materials you will need, and describes how they are physically connected. The software subsection describes how to download the software needed to run the lab.

### 9.2.1 Required Materials

This subsection lists all of the software and hardware materials that you will need to have available before performing this experiment. It also lists any prior experiments that should be performed before running this experiment.

**Hardware**

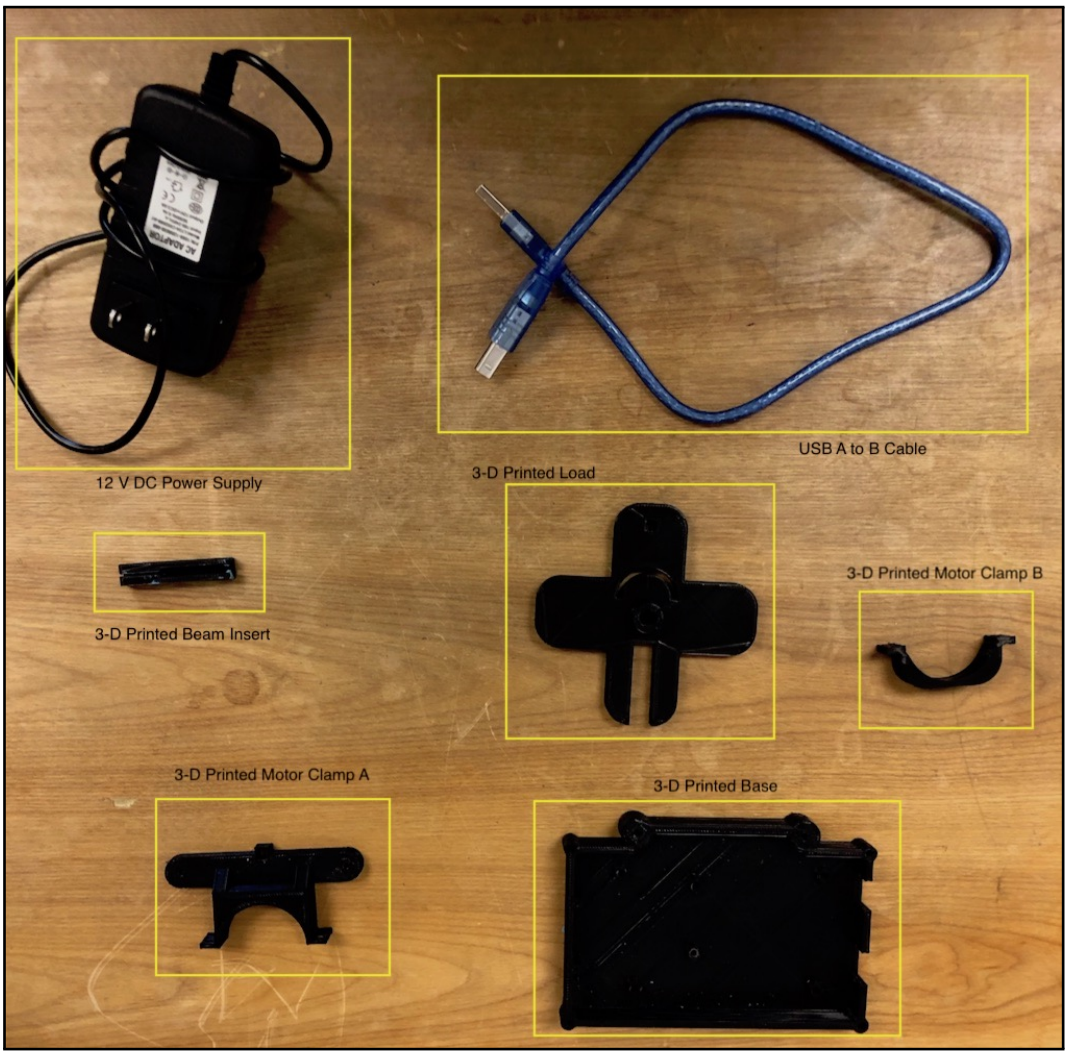



Figure 9.1: Hardware Required for Laboratory

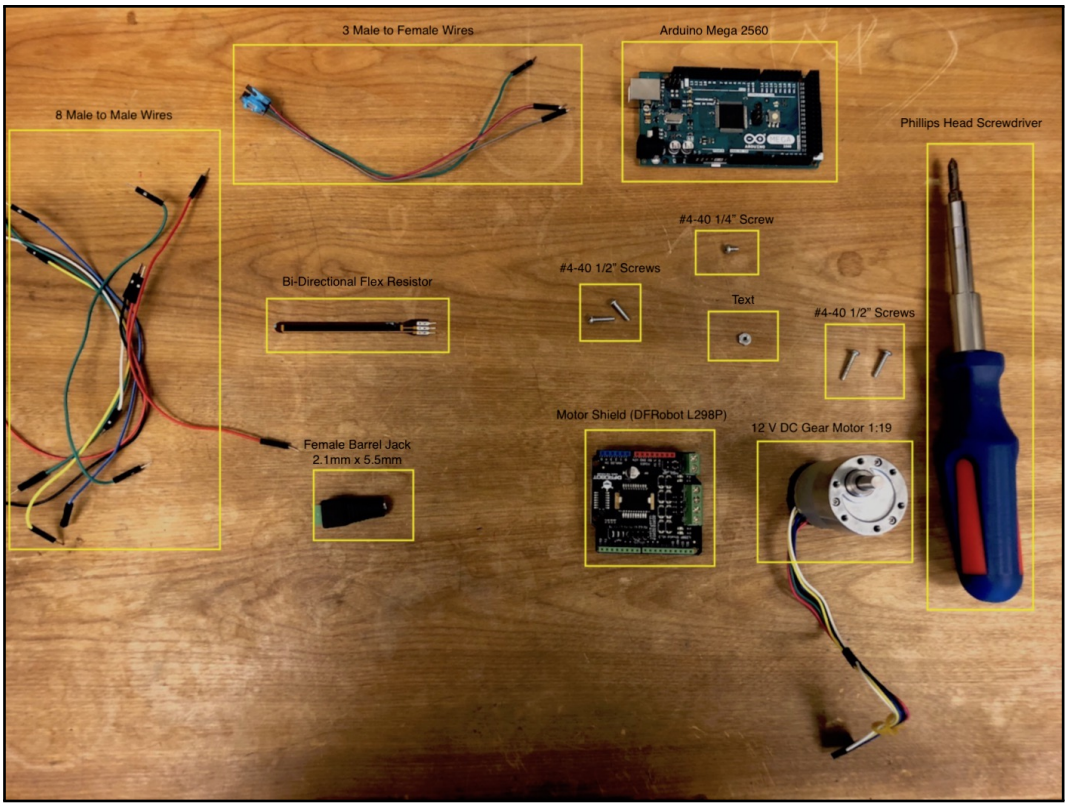



Figure 9.2: Hardware Required for Laboratory

- DC Motor (Pololu item 2822 - Motor with 64 CPR Encoder 19:1 Gearbox)
- Microcontroller (Arduino Mega 2560)
- Motor Shield (DFRobot L298P)
- 12 V DC Power Supply (Universal AC Adapter with 2.1mm x 5.5mm Male Connector)
- USB B to A Converter Cable (USB 2.0 A-Male to B-Male Cable)
- 8 Wires (20cm Male To Male Jumper Wire)
- 3 Wires (20cm Male to Female Jumper Wire)
- Female Barrel Jack (2.1mm x 5.5mm Female CCTV Power Jack Adapter)
- 3" Bidirectional Flexible Bend Sensor

- 1 - # 4-40 1/4" screw
- 2 - # 4-40 1/2" screws
- Screwdriver
- 3-D Printed Beam Insert
- 3-D Printed Base
- 3-D Printed Load
- 3-D Printed Motor Clamp A
- 3-D Printed Motor Clamp B

**Software**

- Matlab/Simulink 2017a
- Windows 10
- All software from the *Open Loop Step Response* and *Closed Loop Step Response* experiments are required for this lab.
- **func_SortSerial.m** (can be obtained in software section on website): put in main path.

**Prerequisite Experiments**

- *Open Loop Step Response*
- *Closed Loop Step Response*

### 9.2.2 Hardware Setup

This section describes the step-by-step hardware setup .

**You can have subsections within the Hardware Setup**

1. Take the # 4-40 1/4" screw and screw the Arduino onto the 3-D printed base as shown in Figure 9.3.

2. Take the motor shield and plug it into the Arduino board as seen in Figure 9.4. Make sure that the pin labeled 5 on the motor shield is plugged into the A5 pin on the Arduino. Additionally, make sure that the Rx pin is aligned with the RX 0 pin on the Arduino. This will ensure that the motor shield is connected properly to the Arduino.

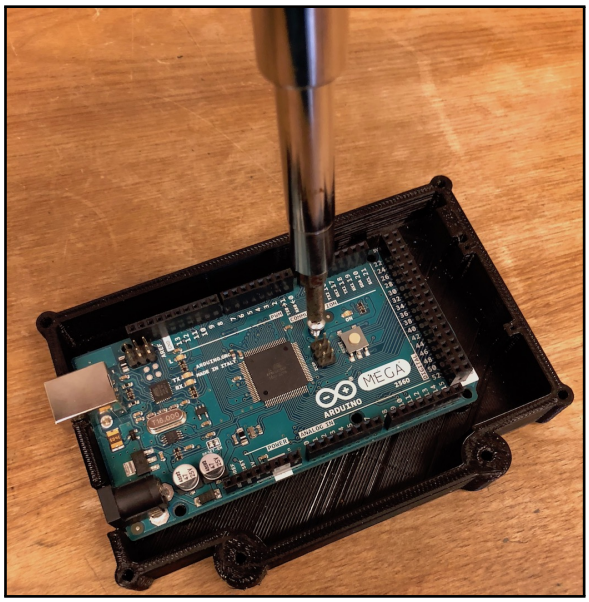



Figure 9.3: Screw On Arduino to 3-D Printed Base

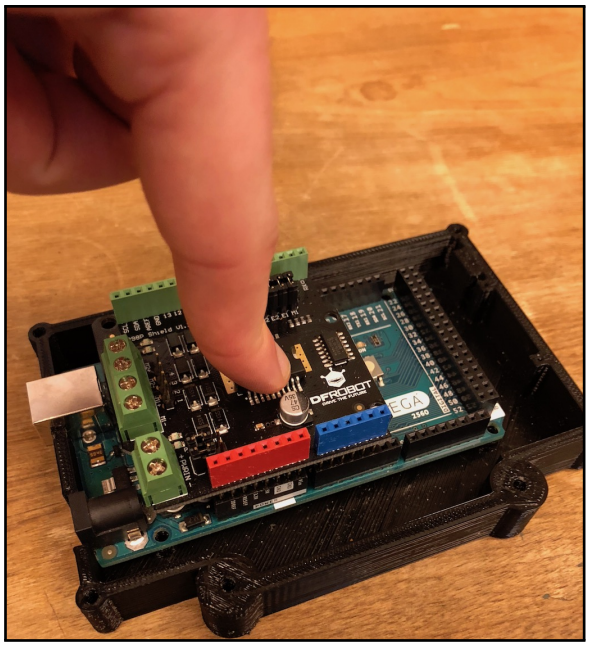


Figure 9.4: Correct Motor Shield Configuration

3. Next connect the motor clamp A to the base with two of the # 4-40 1/2" screws as shown in Figure 9.5.

4. Take the DC motor and place it in the motor clamp A (Figure 9.6).

5. Then, connect the motor clamp B using the other two # 4-40 1/2" screws (Figure 9.7). Make sure that the clamp is snug to eliminate vibration on the motor.

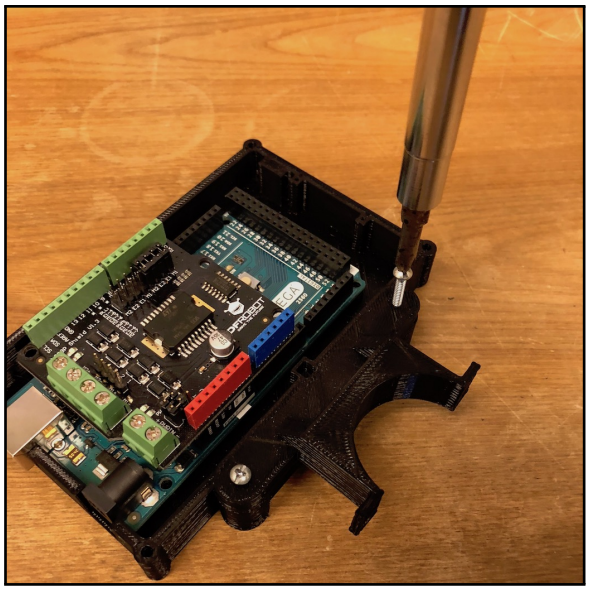


Figure 9.5: Connecting Motor Clamp A

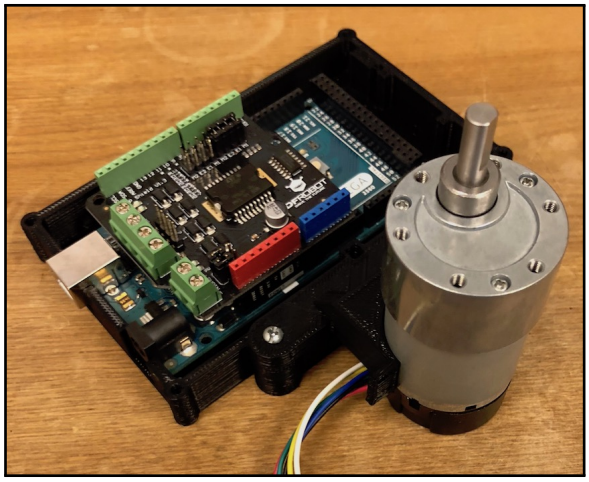


Figure 9.6: Placing Motor

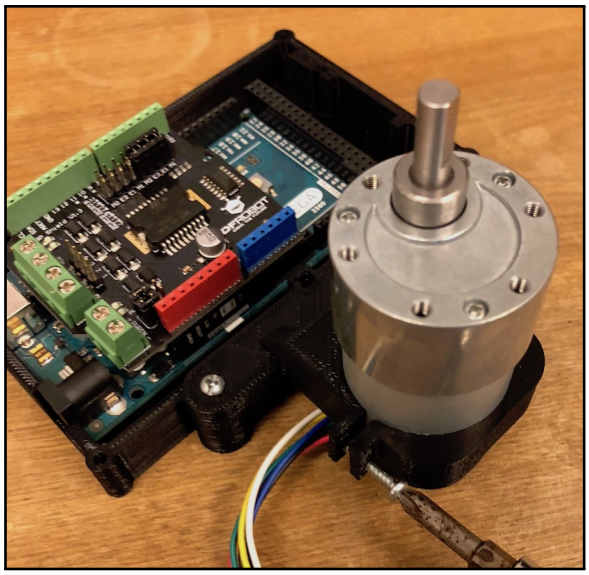


Figure 9.7: Connecting Motor Clamp B

6. Next, take the female barrel jack and screw on a red wire to the positive and a black wire to the negative end as shown Figure 9.8. Each connector port has screw terminals, so clamp them down on the wires by using the screwdriver.

7. Then, connect the other end to the motor shield labeled PWRIN where the positive wire goes to the positive power in and the negative wire goes to the negative power in as shown in Figure 9.9.

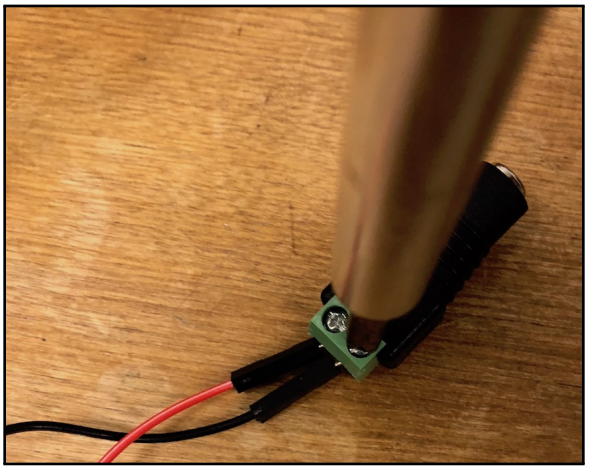


Figure 9.8: Connecting Wires to the Barrel Jack

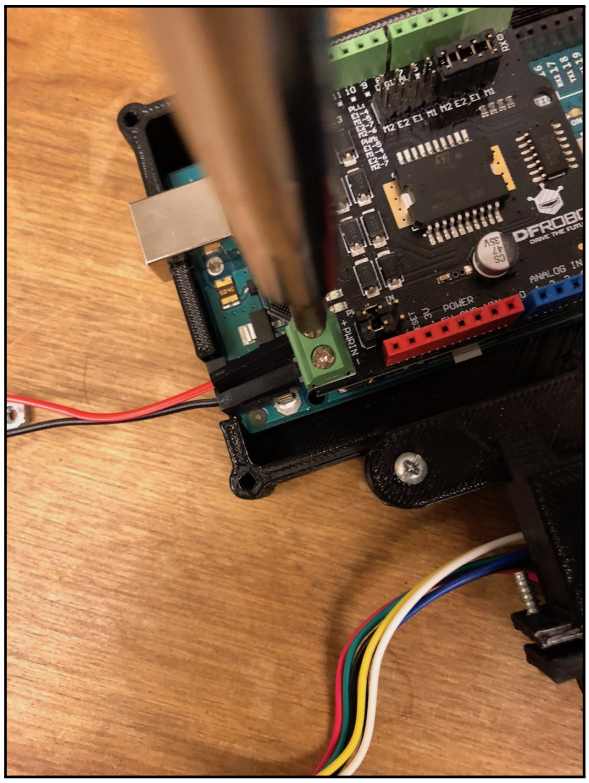



Figure 9.9: Connecting Barrel Jack to Motor Shield

8. Now, take the DC motor wires and connect the other 6 wires into the motor wiring harness as shown in Figure 9.10.

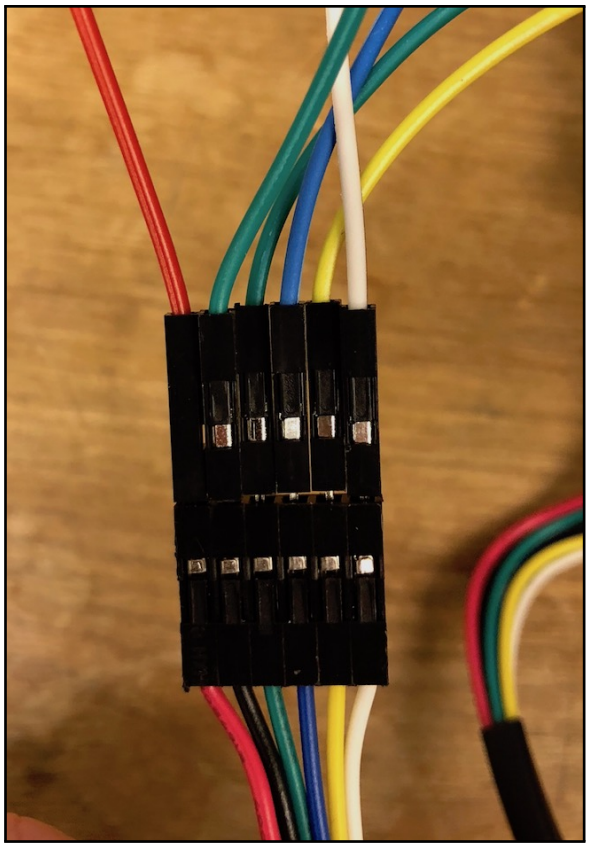


Figure 9.10: Motor Wires

9. Now, take the red and black wires on the motor and connect them to M1+ and M1- on the motor shield respectively as shown in Figure 9.11.

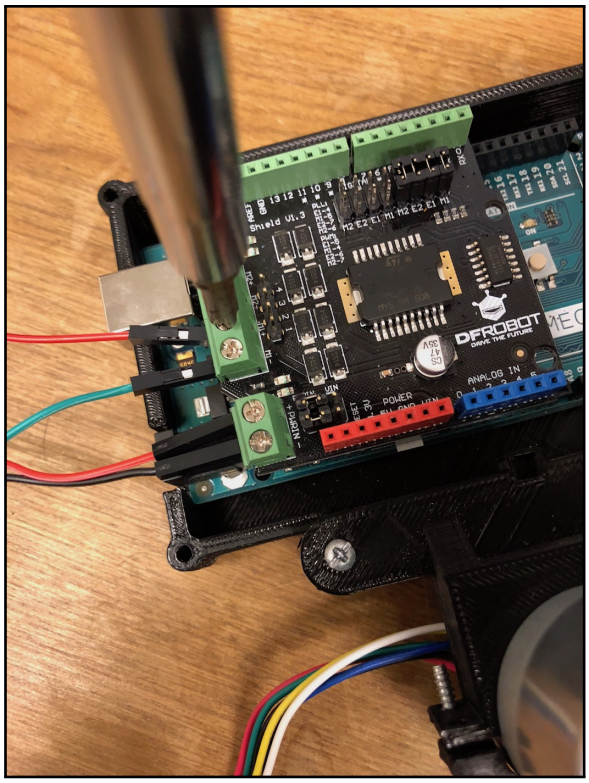



Figure 9.11: Connecting Power to the Motor

10. Next, connect the motor encoder power and ground. Connect the green wire to the GND pin and the blue wire to the 5V pin. connect the encoder wire A (yellow) to pin 2 on the motor shield and the encoder wire B (white) to pin 3. Figure 9.12 shows the final wiring configuration for the encoder. Table 9.1 gives a summary of how the wires need to be connected.

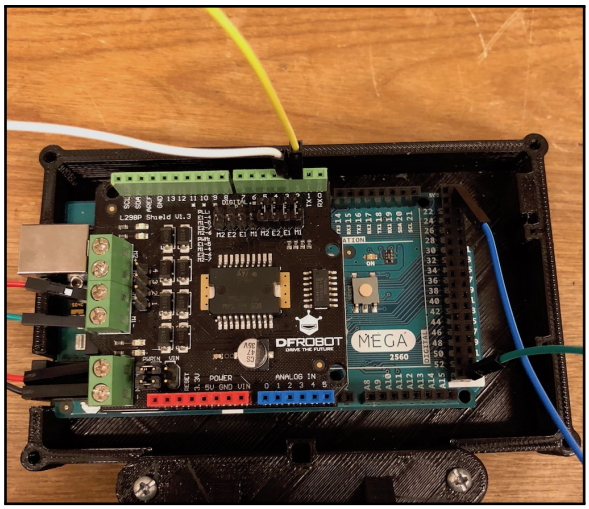



Figure 9.12: Encoder Wiring

Table 9.1: Motor Wires

| Motor Wire | Motor Wire Color | Motor Shield Pin |
|---|---|---|
| Positive Power | Red | M1+ |
| Negative Power | Black | M1- |
| Encoder Power | Blue | 5V |
| Encoder Ground | Green | GND |
| A | Yellow | 2 |
| B | White | 3 |

11. Insert the load onto the motor shaft (Figure 9.13).

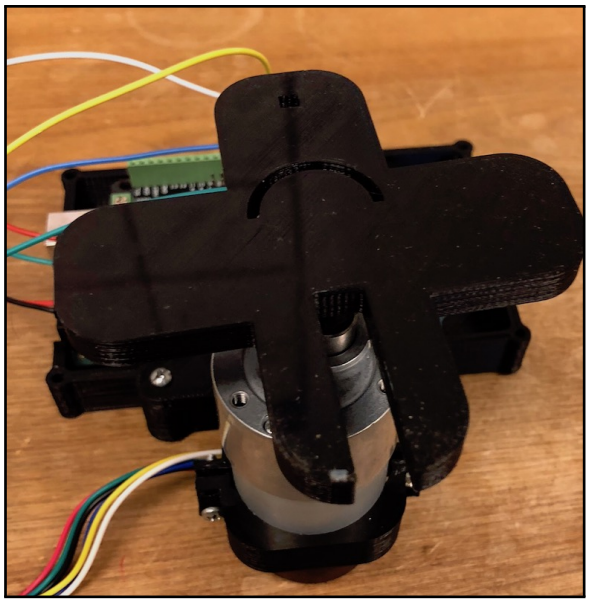


Figure 9.13: Load On Motor

## 9.3 Experimental Procedures Part 1

### 9.3.1 Exercise 1: Deriving Motor Transfer Function

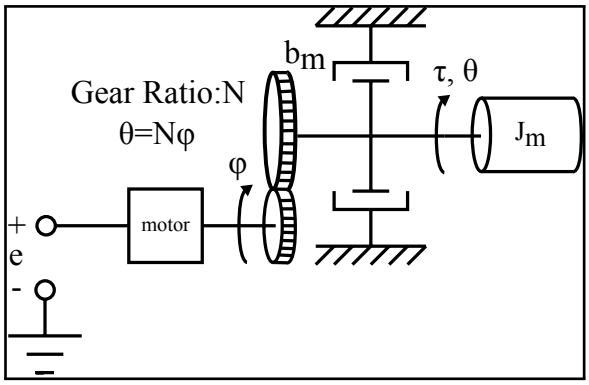



Figure 9.14: Motor Model

Before we assemble the flexible link, we need to find the inertia of the motor, $J_m$, and the viscous friction of the motor, $\beta_m$. The constants can be found by performing the *Open Loop Step Response* and acquiring the transfer function of the motor and load. Since the motor is connected to a gear box, the equations are given by:

$$\frac{J_m}{N}\dot{\omega}(t) = -\frac{\beta_m}{N}\omega(t) + \tau(t) \tag{9.1}$$

$$\tau(t) = \frac{k_t}{R_a}\left[e(t) - \frac{1}{N}k_b\omega(t)\right] \tag{9.2}$$

where $\omega(t)$ is the angular velocity of the load, $\tau(t)$ is the torque, $e(t)$ is the input voltage, $J_m$ is the inertia of the armature and load, $\beta_m$ is the viscous friction of the armature and load, $R_a$ is the armature resistance, $k_t$ is the torque constant of the motor, $k_b$ is the back emf constant of the motor, and $N \leq 1$ is the gear ratio of the gear box.

12. Find the transfer function $G(s) = \frac{\Omega(s)}{E(s)}$ using equations 9.1 and 9.2.

---

- Taking the Laplace transform of both equations and solving:

$$\frac{\Omega(s)}{E(s)} = \frac{k_t N}{R_a J_m s + R_a \beta_m + k_t k_b}$$

---

### 9.3.2 Exercise 2: Theoretical Open Loop Step Response

13. Arrange your transfer function $G(s)$ in the following form

$$G(s) = \frac{K_m}{\tau_m s + 1}$$

14. $K_m$ is the open loop DC gain, and $\tau_m$ is the time constant.

---

- Dividing by the constant term:

$$G(s) = \frac{\frac{k_t N}{R_a\beta_m + k_t k_b}}{\frac{R_a J_m}{R_a\beta_m + k_t k_b}s + 1}$$

---

### 9.3.3 Exercise 3: Experimental Open Loop Step Response

An experimental open loop step response will be run to estimate $J_m$ and $\beta_m$.

15. Open MATLAB 2017a and create a new script named **RFLconstants.m**.

16. The constants used for the motor open loop step response experiment are shown in Figure 9.15. These are the constants that your **RFLconstants.m** script needs to contain.

```
%Sampling Time
Ts = 0.01;

%Gear Ratio (N)
N = 1/18.75;

%Encoder Resolution
rp = 2*pi/64;

%Amplitude of Pulse
amplitude = 4;

%Period of Pulse
period = 6;

%Duty Cycle of Pulse
duty = 50;
```

Figure 9.15: Open Loop Step Response Constants

17. Create a Simulink file and save it as **RFL_OL_exp.slx**.

18. Figure 9.16 shows what the Simulink file should look like. Figure 9.17 contains the motor subsystem block. **Note:** Please refer to *Simple DC motor*, *Sampling and Data Acquisition*, and *Open Loop Step Response* experiments for all details regarding running on target hardware, setting up THL libraries, and using the serial plot.

19. The pulse generator block contains the fields amplitude, period and duty cycle. Enter the names amplitude, period, and duty respectively, in these fields.

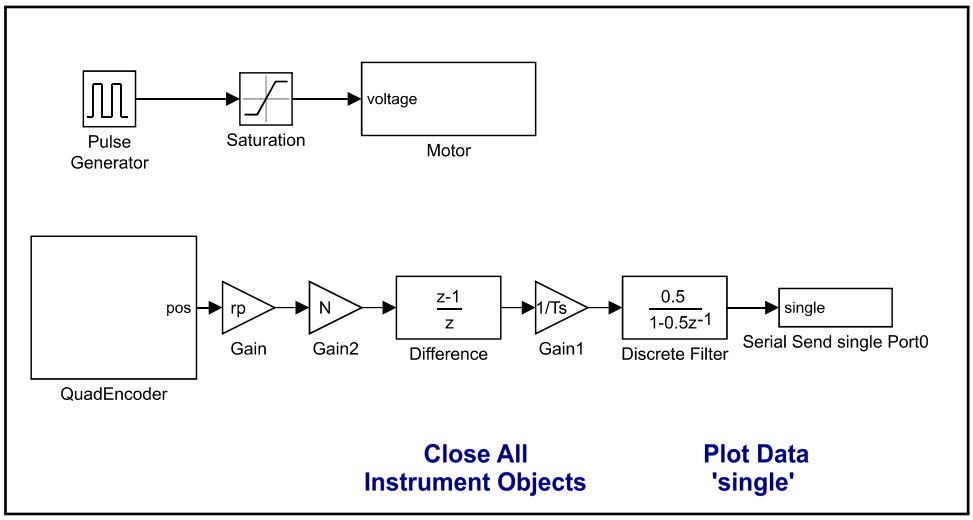



Figure 9.16: Open Loop Step Response Model

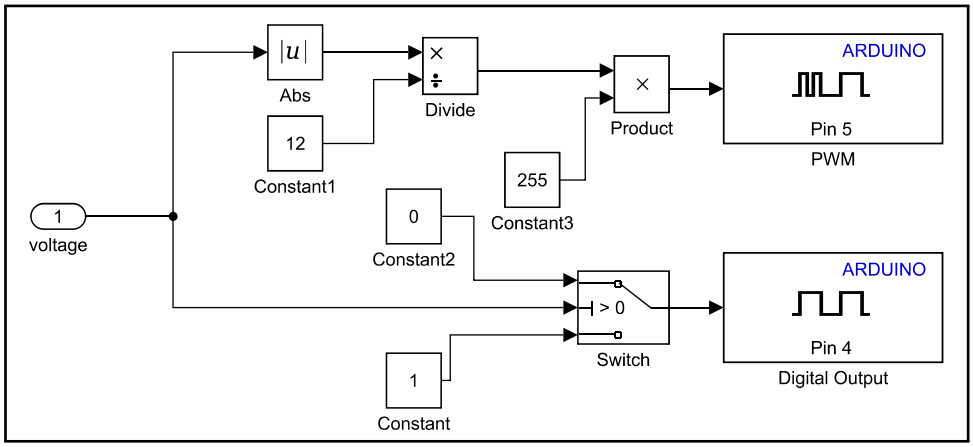



Figure 9.17: Motor Subsystem Block

20. Perform the Open Loop Step Response Experiment to obtain an experimental open loop step response for your motor and load.

21. Find $K_m$ and $\tau_m$ using the step response plot.

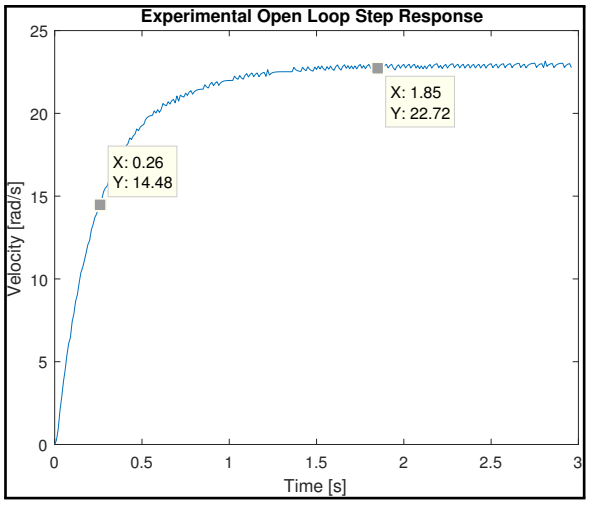



Figure 9.18: Experimental Open Loop Step Response Plot

- $K_m = 22.72$
- $\tau_m = 0.26$

---

### 9.3.4 Exercise 4: Simulation of Open Loop Step Response and Comparison

22. Simulate the open loop step response using $K_m$ and $\tau_m$ found in the previous step. Make sure your experiment and simulation match to get good results.

23. Solve for $J_m$ and $\beta_m$ using the constants given:

$$k_t = 0.0058 \frac{Nm}{A}$$

$$k_b = 0.0058 \frac{\frac{V}{Rad}}{s}$$

$$R_a = 2.6\Omega$$

$$N = \frac{1}{18.75}$$

---

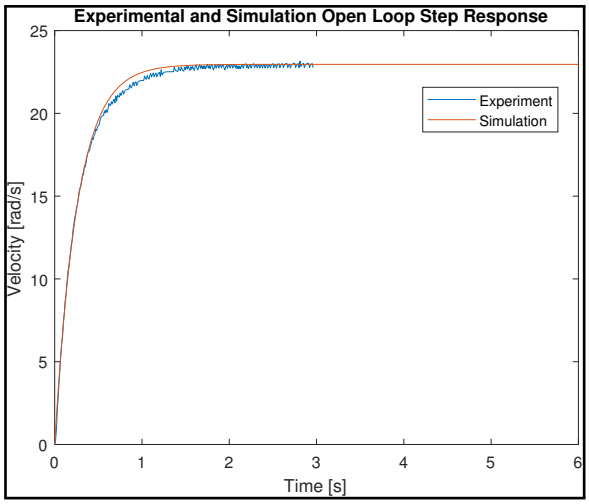



Figure 9.19: Experiment and Simulation Open Loop Step Response Plot

- $\beta_m = \frac{Nk_b - Nk_bk_t}{K_mR_a} = 8.0205 \times 10^{-6}$
- $J_m = \frac{\tau_m(R_a\beta_m + k_tk_b)}{R_a} = 5.11 \times 10^{-6}$

---

### 9.3.5 Exercise 5: Theory - Optimal Motor Position Control

This section you will derive an optimal control for motor position. To derive the control gains in MATLAB you will need a state space representation of your system. The input to the system will be in the form $u = K(x_d - x)$ where $K$ is the control gains, $x_d$ is the desired state (position and velocity), and $x$ is the state. The equations of motion are

$$\frac{J_m}{N}\ddot{\theta}(t) = -\frac{\beta_m}{N}\dot{\theta}(t) + \tau(t) \tag{9.3}$$

$$\tau(t) = \frac{k_t}{R_a}\left[e(t) - \frac{1}{N}k_b\dot{\theta}(t)\right] \tag{9.4}$$

24. Derive the state space model using equations 9.3 and 9.4, using the definitions $x_1 = \theta$ and $x_2 = \dot{\theta}$.

---

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{J_m}\left(\beta_m + \frac{k_t k_b}{R_a}\right) \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{N k_t}{J_m R_a} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x$$

---

### 9.3.6 Exercise 6: Simulation - Optimal Motor Position Control

25. Assuming that there is no noise in the system, and assuming all states are measurable, use the "lqr" command in MATLAB to find the optimal control gains. You will need to set the weighting matrices in the performance index. In MATLAB type **K_m1=lqr(A,B,Q,R)**. Make sure all of your variables are defined. A should be a square matrix, B should be a vector. Q is a square matrix the same size as A. and R is a scalar. Q is weighting matrix that penalizes the states and R is a value that penalizes energy used.

---

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$R = 0.001$$

$$K_{m1} = \begin{bmatrix} 31.6228 & 1.5273 \end{bmatrix}$$

---

26. To simulate the control, create a Simulink file named **RFL_partialControl_sim.slx**. The file should look like Figure 9.20.

27. The saturation block is set to 12 and -12 (the voltage of the power supply).

28. Make sure everything is set correctly in the state space block. The initial conditions should be set to [0;0].

29. xd is the desired position and velocity. Set this to [pi/2;0].

30. Make sure the gain matrix is set to perform matrix calculations.

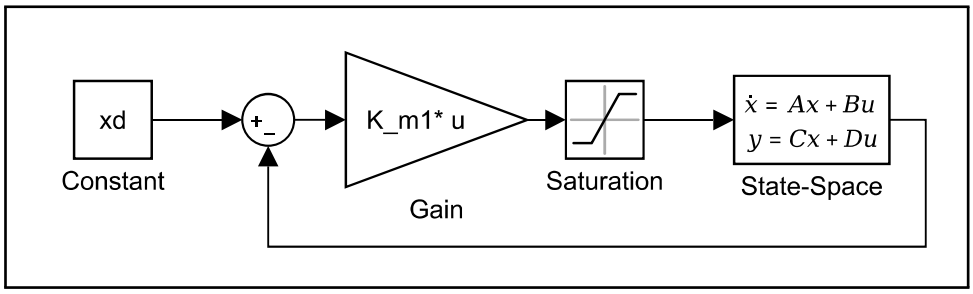



Figure 9.20: Simulation File For Optimal Motor Control

31. Simulate the file for 5 seconds and plot voltage, $x_1$, and $x_2$.

---

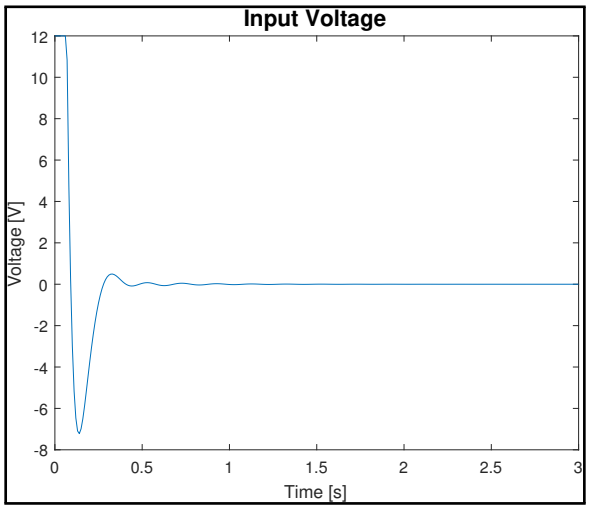



Figure 9.21: Exercise 6: Simulated Voltage

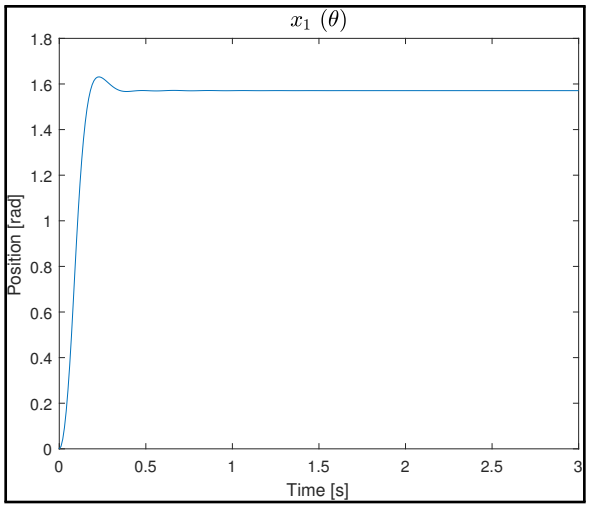



Figure 9.22: Exercise 6: Simulated Position

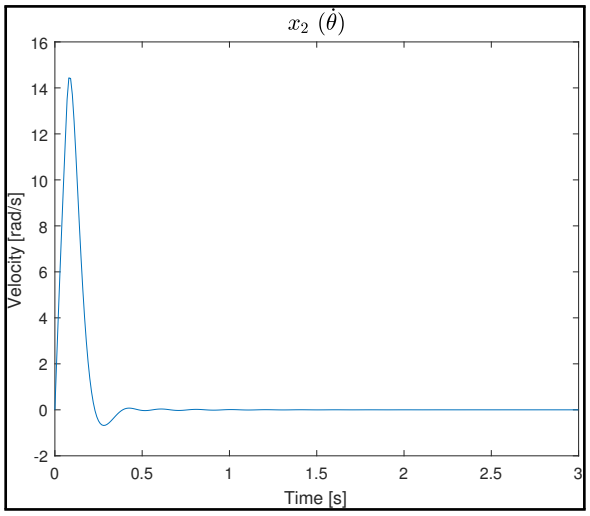



Figure 9.23: Exercise 6: Simulated Velocity

---

32. Is the response what you expected? If not, manipulate Q and R to give a fast response with little overshoot.

---

- The response desired is a fast response with slight overshoot. Q was selected to do so.

---

33. Continue to adjust Q and/or R until you obtain a desired response for the motor load.

### 9.3.7 Exercise 7: Experiment - Optimal Motor Position Control

34. Create another Simulink file named **RFL_partialControl_exp.slx**. The file should look like Figure 9.24.

35. The pulse generator block amplitude needs to be set to pi/2 and the period set to 10 seconds.

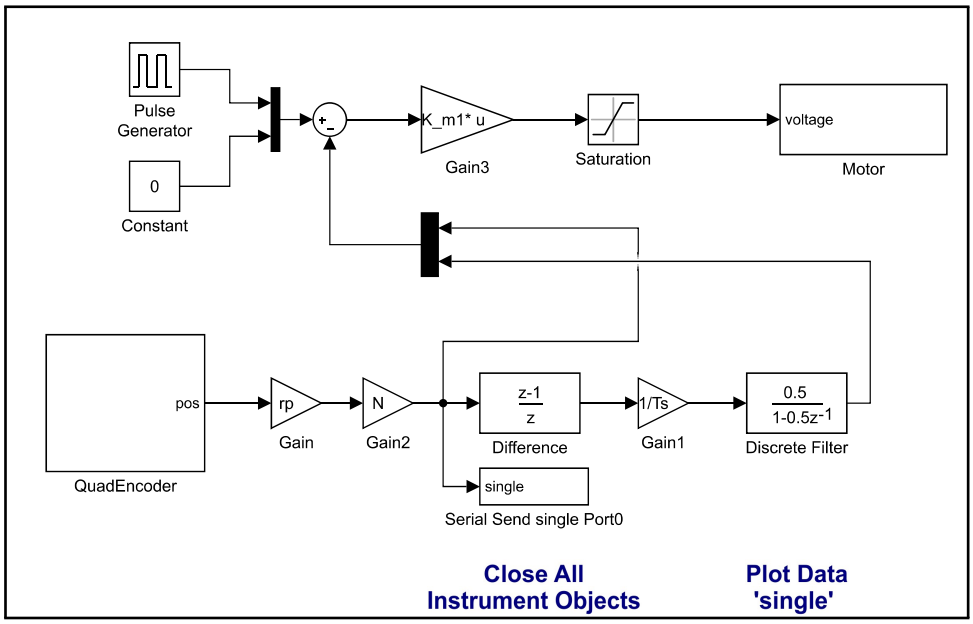



Figure 9.24: Experiment File For Optimal Motor Control

36. Run the experiment and compare the simulation to the experiment. How closely do they match? What could cause any differences.

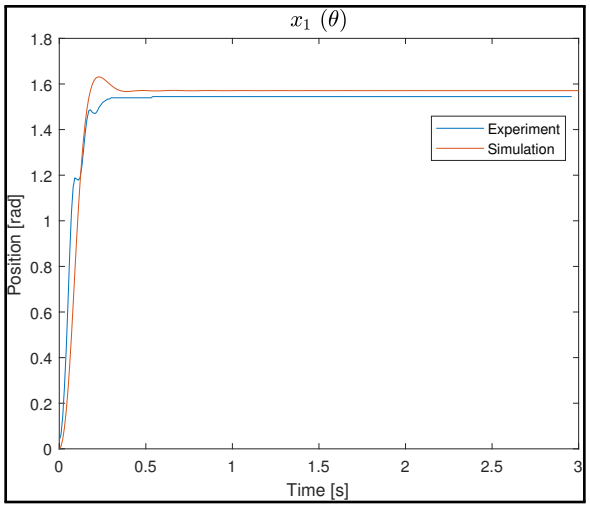



Figure 9.25: Exercise 7: Experimental and Simulated Position

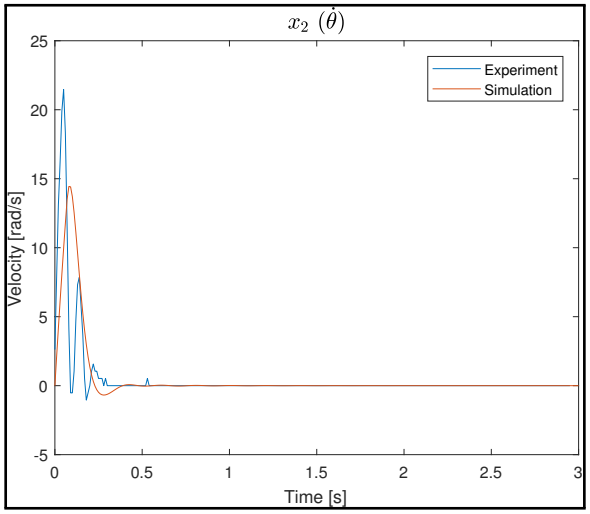



Figure 9.26: Exercise 7: Experimental and Simulated Velocity

- In the position plot the rise times are closely related but there is no overshoot in the experiment. The differences could be caused by the nonlinearities in the motor. At the beginning of the simulation the voltage also saturates for a couple of milliseconds, which could cause some differences?

---

37. You may want to experiment with different control gains if they do match properly.

## 9.4 Hardware Setup Part 2

This section describes how to set up the flexible link.

### 9.4.1 Attaching Flexible Link

38. Place a small ball of sticky tack onto the end of the flex resistor (Figure 9.27)

Figure 9.27: Adding a Point Mass to Link

39. Strip three male-to-female wires on the female end (See Figure 9.28 for an example)

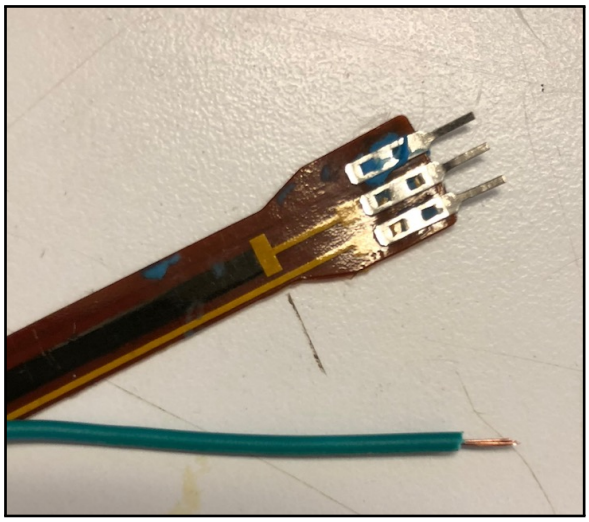


Figure 9.28: stripped Wire

40. Solder all three wires to the pins of the flex resistor.

41. Insert the flex resistor into the 3-D printed beam insert and secure with sticky tack as shown in Figure 9.29.

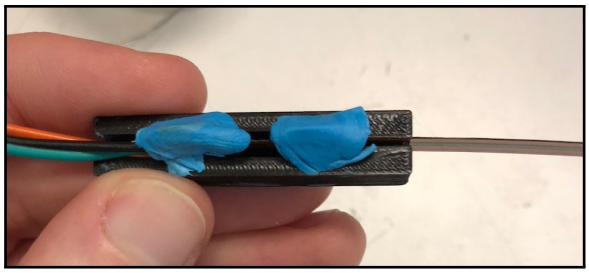


Figure 9.29: Inserting Sensor

42. Place the insert into the load, and set the system so that the link lines up to the 90 degree of the protractor in Figure 9.30. The two outer wires of the sensor go to 5V and Ground, and the middle wires goes to the Analog 2 pin (similar to a potentiometer).

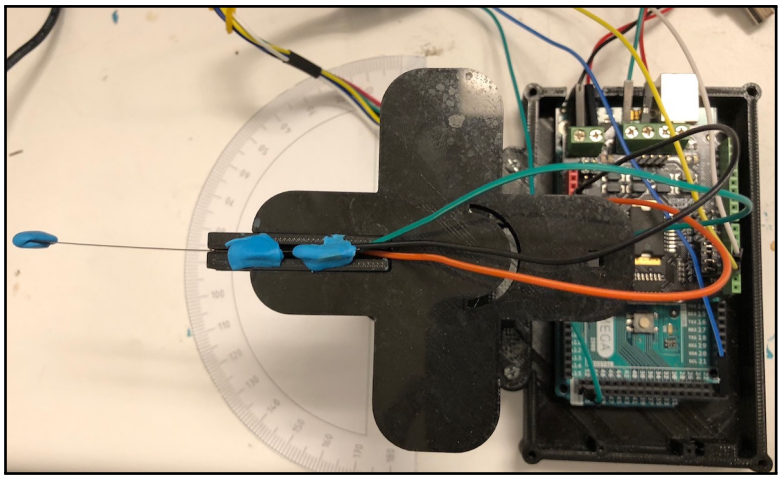


Figure 9.30: Overall System

### 9.4.2 Calibrating Flexible Link

43. To calibrate the beam we first want to observe the voltage with the beam at rest. Figure 9.31 shows a file that can be set up to observe this value. The analog to digital converter on the Arduino is a 10 bit value (0-1023). Your value will be somewhere around 510. To read the value, open the Plot data 'single' window and zoom the figure to get an accurate reading.

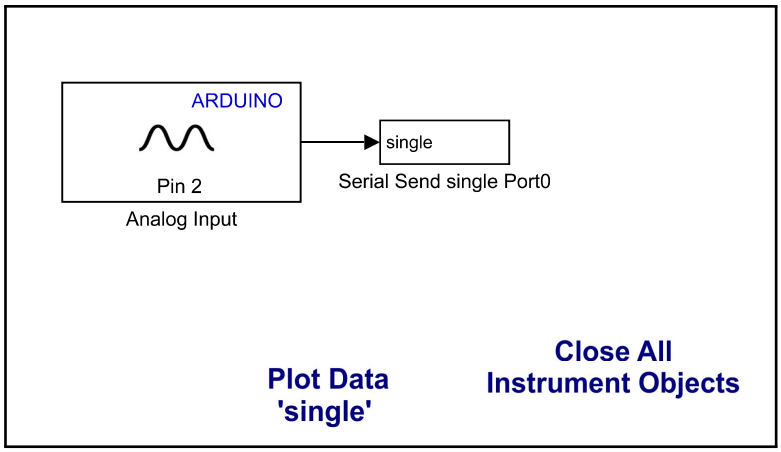



Figure 9.31: Beam Calibration File

44. Make note of what the value is.

---

- The hardware will be different due to tolerances. The resting value obtained for the experiment is 516.

---

45. Next we need to map voltages to angles in degrees. Using the protractor as a guide, bend the beam to a specific angle and read the voltage from the plot.

46. Subtract off the resting value from your voltage and take note of the angle and the voltage. Repeat for as many angles as you think necessary. Make sure to do both positive and negative angles as the flex resistor behavior is not symmetric. We will use a lookup table to make the conversion. Figure 9.32 shows an example of what the lookup table should look like. For this example, angles of 0, 50, and -50 degrees were measured. **Note** that the zero angle will always have a voltage of zero, since you are subtracting off the resting voltage.

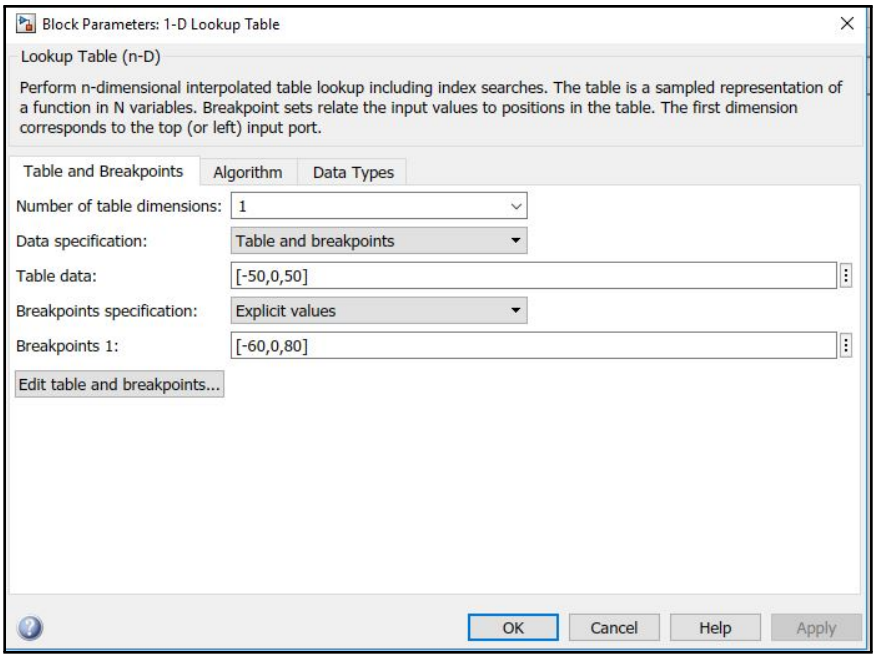



Figure 9.32: 1-D Lookup Table

47. Your final flex resistor subsystem will look something similar to Figure 9.33. The final measurement will need to be converted to radians. **Note** that we are subtracting off the resting voltage. Make sure that **resting_value** is defined in the workspace before running the model.

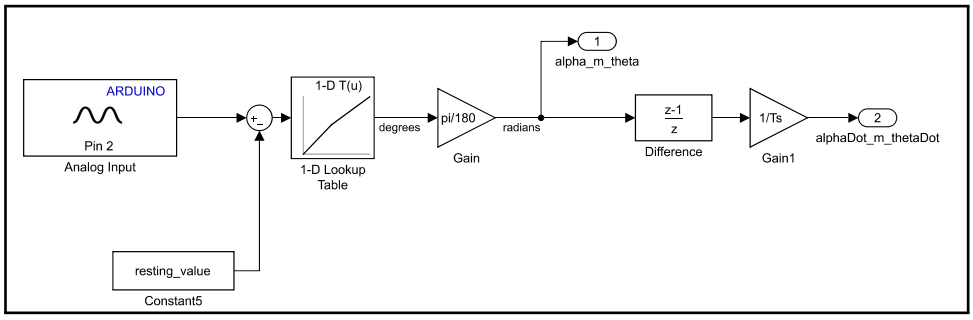



Figure 9.33: Link Sensor Subsystem

## 9.5 Experimental Procedures Part 2

In this section we will derive the dynamics of the link, then using that knowledge to design an optimal control system (Full State Feedback) to minimizes the link deflections. We will then be able to make comparisons with the optimal control for the motor and

load (Partial State Feedback).

### 9.5.1 Rotary Flexible Link Model

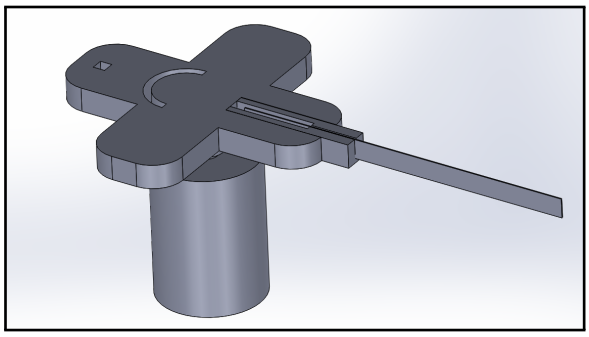


Figure 9.34: Isometric View of Rotary Flexible Link

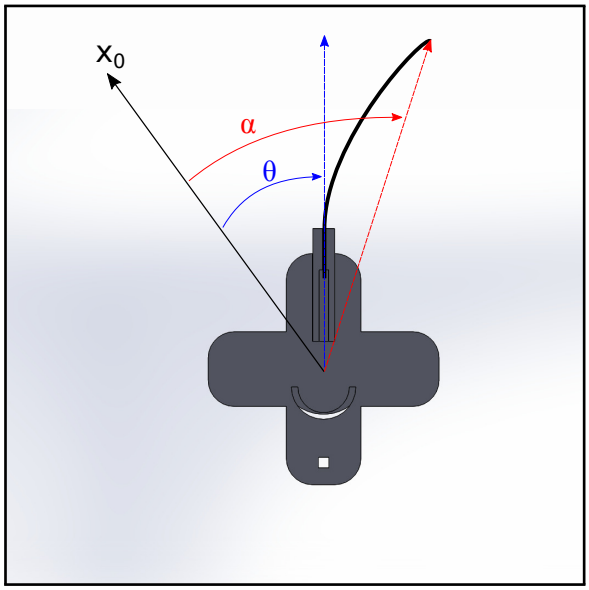



Figure 9.35: Angle Definitions

As shown in Figure 9.35, $\theta$ is the measured angle of the load from the starting point and $\alpha$ is $\theta$ plus the link deflection. The flexible link can be seen as a spring mass damper system. Figure 9.36 shows a representation of the total rotary flexible link system.

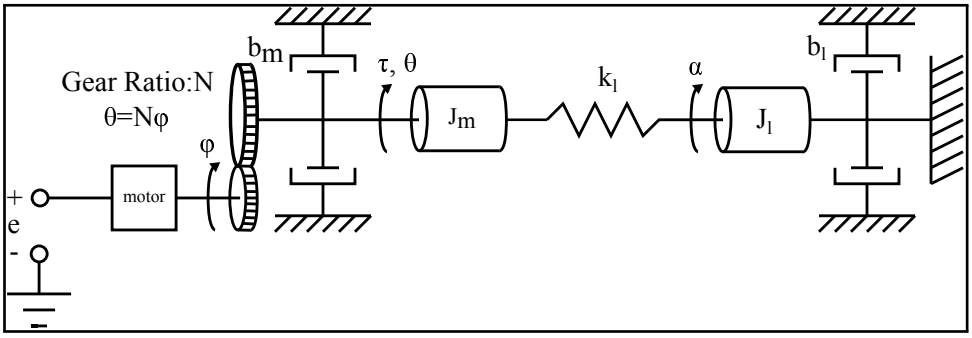



Figure 9.36: Motor Model

The equations of motion for the Rotary Flexible Link are

$$\frac{J_m}{N}\ddot{\theta}(t) = -\frac{\beta_m}{N}\dot{\theta}(t) - k_l\big(\theta(t) - \alpha(t)\big) + \tau(t) \tag{9.5}$$

$$J_l\ddot{\alpha}(t) = -k_l\big(\alpha(t) - \theta(t)\big) - \beta_l\dot{\alpha}(t) \tag{9.6}$$

$$\tau(t) = \frac{k_t}{R_a}\big[e(t) - \frac{1}{N}k_b\dot{\theta}(t)\big] \tag{9.7}$$

where $J_l$ is the inertia of the link, $k_l$ is the spring constant of the link, and $\beta_l$ is the damping coefficient of the link.

$$k_t = 0.0058\frac{Nm}{A}$$

$$k_b = 0.0058\frac{\frac{V}{Rad}}{s}$$

$$R_a = 2.6\Omega$$

$$N = \frac{1}{18.75}$$

$$J_l \approx 1.26 \times 10^{-6} kg \times m^2$$

**Note:** The inertia of link, $J_l$ may not be exact due to the sticky tack weight on the point mass.

48. Create a state space model using Equations 9.5-9.7, where $x_1 = \theta$, $x_2 = \dot{\theta}$, $x_3 = \alpha$, $x_4 = \dot{\alpha}$.

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{Nk_l}{J_m} & -\frac{1}{J_m}\left(\beta_m + \frac{k_t k_b}{R_a}\right) & \frac{Nk_l}{J_m} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_l}{J_l} & 0 & -\frac{k_l}{J_l} & -\frac{\beta_l}{J_l} \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{Nk_t}{J_m R_a} \\ 0 \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x$$

---

### 9.5.2 Exercise 8: Dynamics of Flexible Link

To derive $k_l$ and $\beta_l$ we can assume $\theta$ is fixed at zero. Let the initial condition of $\alpha$ be $\alpha(0) = \alpha_0$. Thus we are left with the Equation 9.8.

$$\ddot{\alpha} + \frac{\beta_l}{J_l}\dot{\alpha} + \frac{k_l}{J_l}\alpha = 0 \tag{9.8}$$

Figure 9.37 shows the dynamics of the link when set at some initial condition. There are two points of concern ($t_1$,c) and ($t_2$,d). We can use the knowledge of system dynamics to derive the pole locations (a±jb) and result in deriving, experimentally, $k_l$ and $\beta_l$.

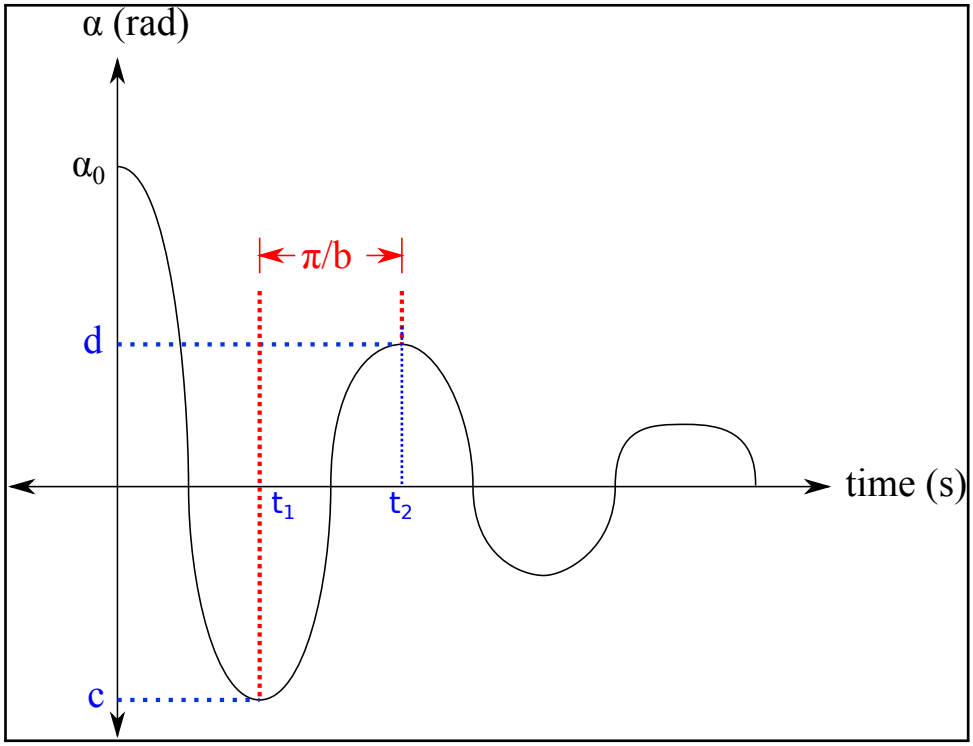



Figure 9.37: Oscillation of Flexible Link For Fixed Motor Position

49. Recreate the plot shown in Figure 9.37 experimentally with your link. Take note of the initial condition used in radians.

---

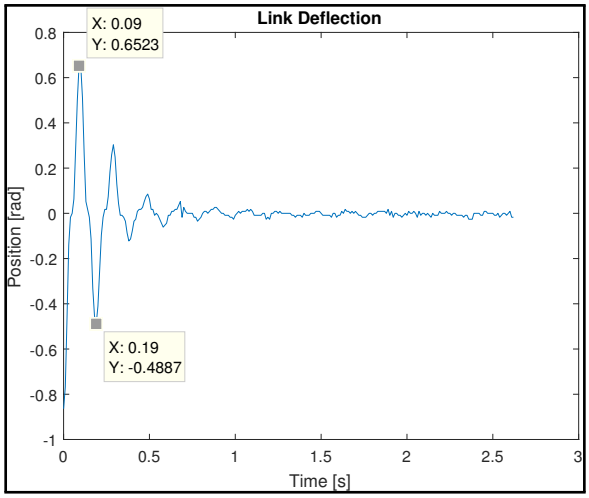



Figure 9.38: Link Deflection

---

50. Use equations 9.9 and 9.10 to calculate the real and imaginary parts of the pole locations.

$$\frac{\pi}{b} = t_2 - t_1 \tag{9.9}$$

$$\frac{|d|}{|c|} = e^{\frac{-a\pi}{b}} \tag{9.10}$$

---

$$t_1 = 0.09$$

$$t_2 = 0.19$$

$$c = 0.6523$$

$$d = -0.4887$$

$$b = 31.4159$$

$$a = 2.8876$$

---

51. Using the pole locations, write the characteristic polynomial in the form $s^2 + xs + y = 0$. Notice if you take the Laplace Transform of Equation 9.8, you get the same form. ($x = \frac{\beta_l}{J_l}$ and $y = \frac{k_l}{J_l}$)

52. Compute $k_l$ and $\beta_l$.

---

$$s^2 + 5.7751s + 995.2984 = 0$$

$$\frac{\beta_l}{J_l} = 5.7751 \Rightarrow \beta_l = 7.2791 \times 10^{-6}$$
$$\frac{k_l}{J_l} = 995.2984 \Rightarrow k_l = 0.0013$$

---

### 9.5.3 Exercise 9: Simulation - Partial State Feedback

53. Create a simulation file similar to **RFL_partialControl_sim.slx**, but with the A, B, C, and D matrices derived in step 48.

54. Now make K_motor = [K_m1,0,0], where K_m1 was found in step 25 (See Figure 9.39). This represents the partial state feedback, where $x_3$ and $x_4$ are measured, but not controlled. The controller is designed as if the beam is stiff, but the flexible link is simulated.

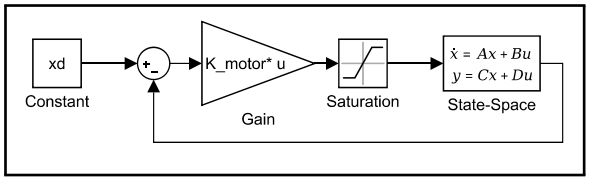



Figure 9.39: Simulation File For Step 54

55. Make the initial conditions of the State Space block [0;0;0;0].

56. Make xd = [pi/2;0;pi/2;0].

57. Run the simulation file for 5 seconds and plot voltage and $x_1$-$x_4$.

58. Discuss any observations made on all four states, as well as the voltage.

---

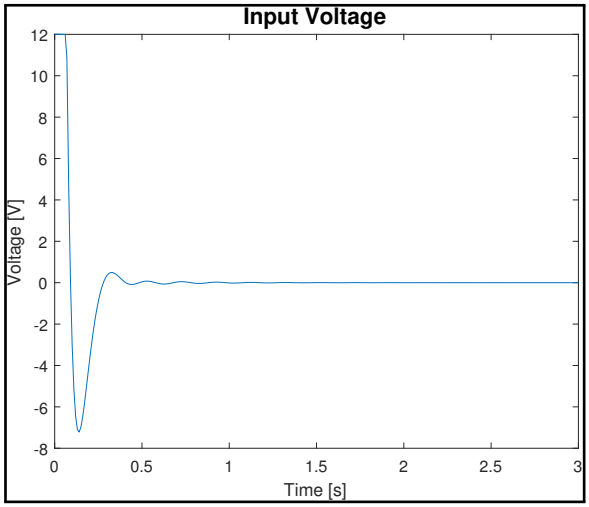



Figure 9.40: Simulated Input Voltage

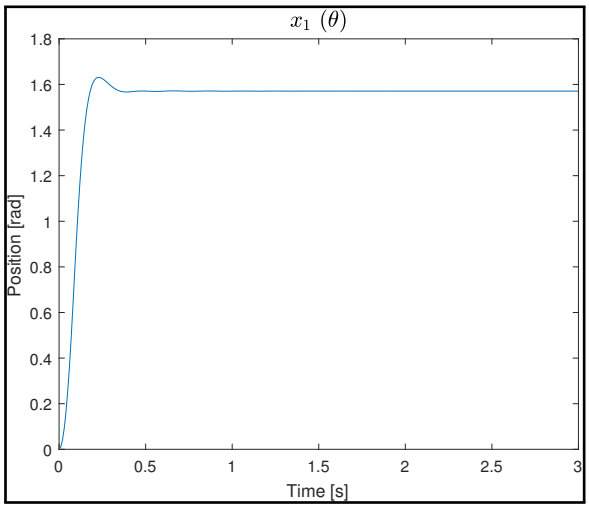



Figure 9.41: Simulated Load Position

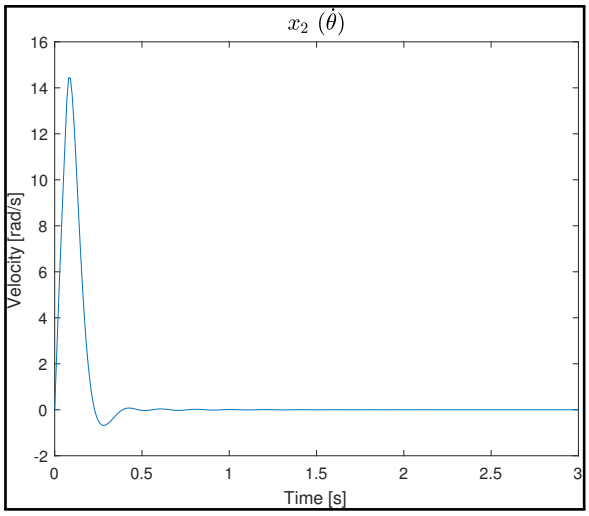



Figure 9.42: Simulated Load Velocity

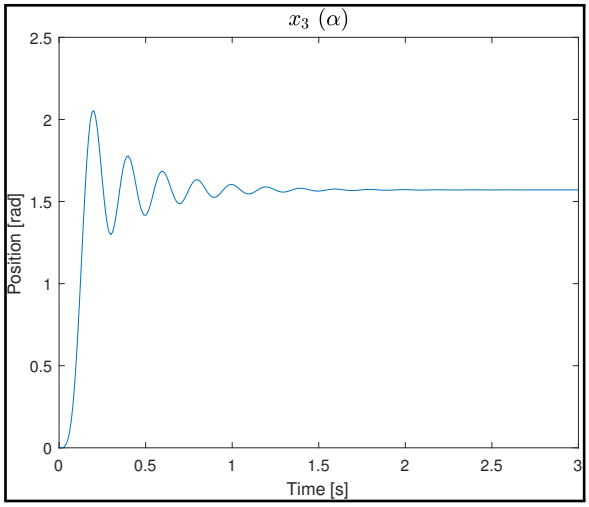



Figure 9.43: Simulated Load Plus Link Deflection Position

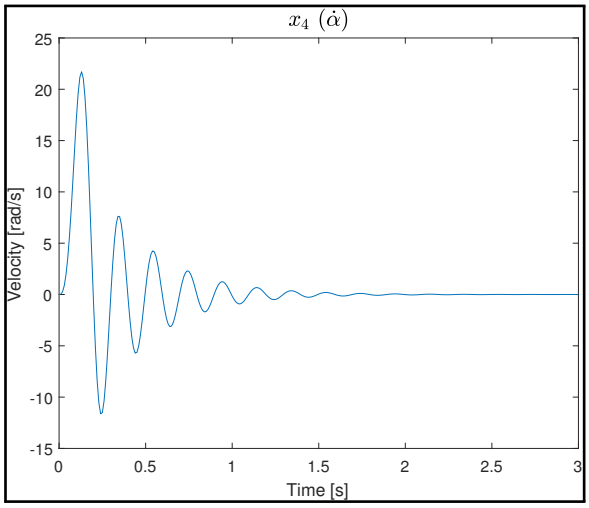



Figure 9.44: Simulated Load Plus Link Deflection Velocity

- Since the feedback gains on the states $x_3$ and $x_4$ are 0, then those states are not controlled. Therefore the beam is very under-damped and has a fair amount of oscillations. It appears that there is a very small voltage around 0.3 seconds, this voltage will most likely not be able to overcome the static friction in the motor.

---

### 9.5.4 Exercise 10: Experiment - Partial State Feedback

59. Create a new Simulink file and save it as **RFL_exp.slx**.

60. Figure 9.45 shows the overall Simulink model. Figure 9.46 shows the plant (Rotary Flexible Link). The Flexible_Link subsystem is for the flex resistor sensor system. Figure 9.33 shows how it should be arranged. **Note:** that we will be able to switch between the controller designed for the 2nd order system (without considering the flexible link) and the full state feedback controller for the 4th order system.

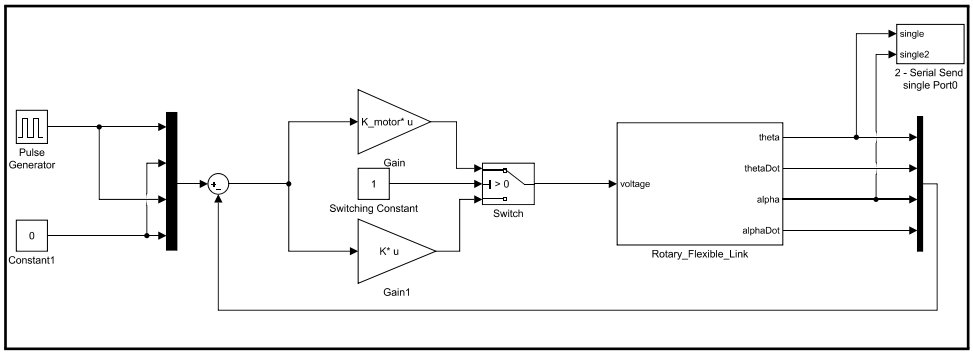



Figure 9.45: Overall Experiment File

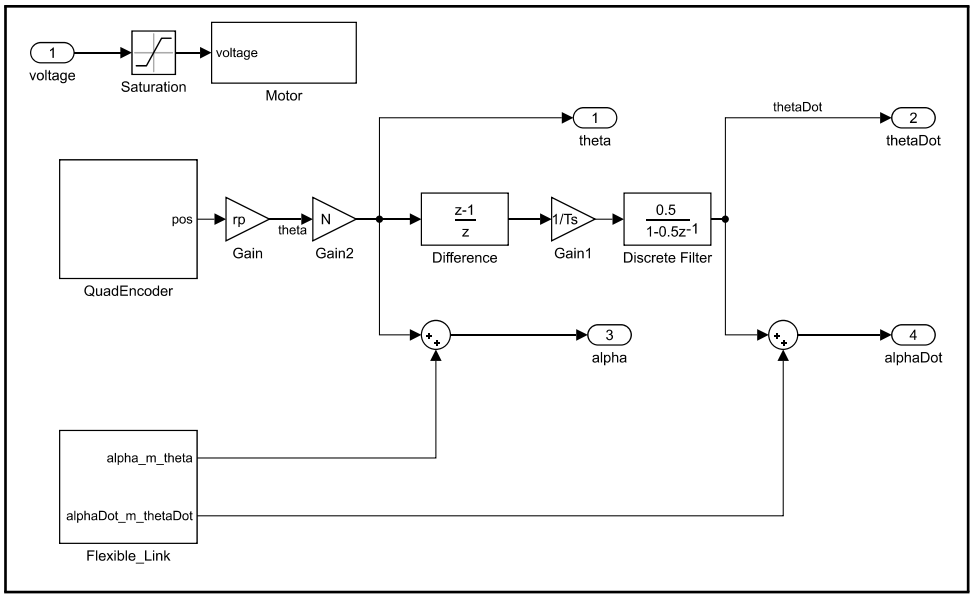



Figure 9.46: Plant

61. The pulse generator block amplitude is set to pi/2, the period is set to 5, and the duty cycle is set to 50.

62. Run the experiment and verify that your simulation matches your experiment.

63. WindowDat is the variable saved for data. To sort the data,
    call [d1,d2] = func_SortSerial(WindowDat) to separate $\alpha$ and $\theta$. **Note:** It should be easy to associate d1 and d2 to $\alpha$ and $\theta$ due to the oscillations of the link.

64. What differences do you see between simulation and experiment?

65. What are the reasons why they might not match exactly?

66. You may need to re calibrate your beam if they do not match closely.

---

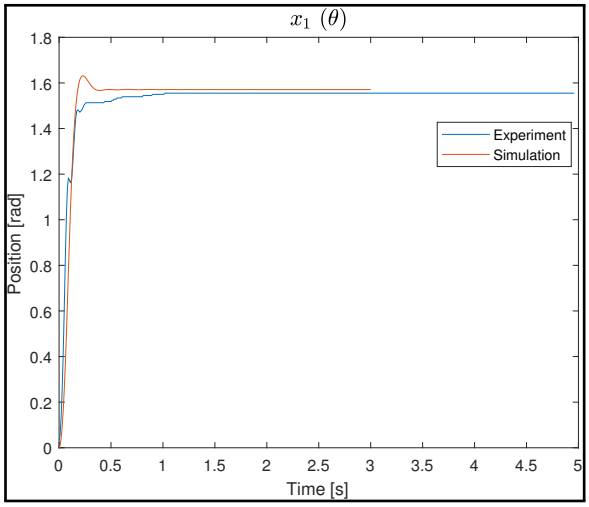



Figure 9.47: Experimental and Simulated Load Position

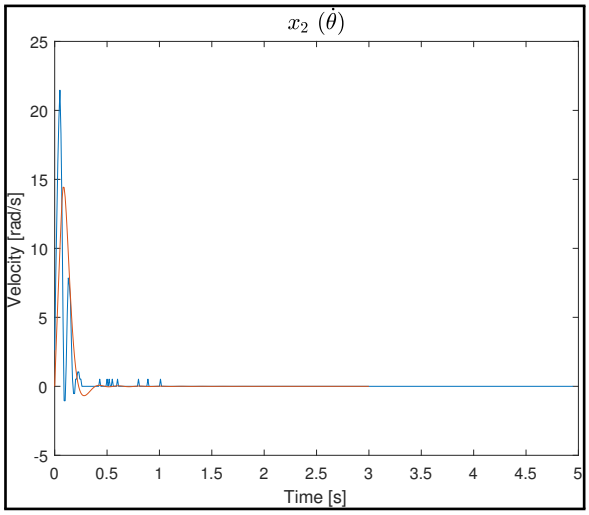



Figure 9.48: Experimental and Simulated Load Velocity

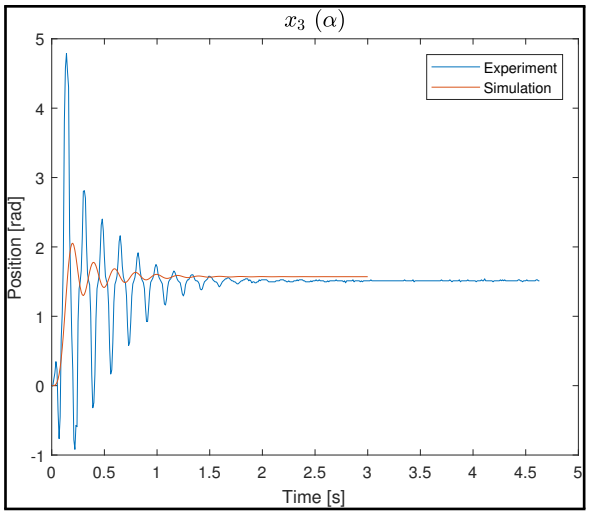



Figure 9.49: Experimental and Simulated Load Plus Link Deflection Position

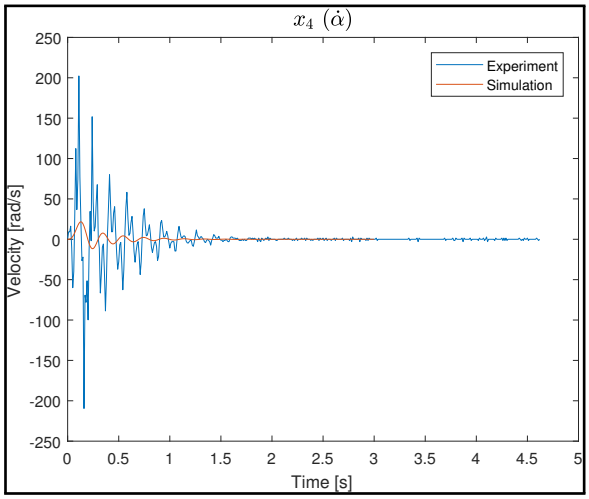



Figure 9.50: Experimental and Simulated Load Plus Link Deflection Velocity

- The first state to discuss is $x_1$. The rise time on the experiment closely matches the simulation. However the overshoot does not appear in the experiment like it does in the simulation. This might have happened due to the static friction in the motor. It shows that there is a small amount of steady state error.

- As for $x_2$, the experiment and simulation have fairly similar shapes overall.

- $x_3$ experiment and simulation do not entirely match up. The overshoot and frequency of oscillation are not in sync. This could be due to the calibration of the flexible link. We might need to take more measurements at more angles. There is

a deadband at the resting position of the position sensor. The sensor appears to not be linear and therefore, in the future may need to be modified to compensate.

- Because $x_3$ does not match, $x_4$ will not match either.

---

### 9.5.5 Exercise 11: Theory - Full State Feedback Optimal Control

Now we would like to design an optimal state variable feedback controller for all four states to minimize oscillations in the flexible link. We will design a control input $u = K(x_d - x)$ to get a fast response with little oscillations on the flexible link. The Q and R matrix will need to be designed such that this is achieved. To choose a proper Q, you may want to get creative in selecting the off diagonals, primarily ones that pertain to $x_1$ and $x_3$.

### 9.5.6 Exercise 12: Simulation - Full State Feedback Optimal Control

67. Obtain control gains using the "lqr" command in MATLAB, using the 4th order system model.

---

$$Q = \begin{bmatrix} 5 & 0 & -1 & 0 \\ 0 & 0.05 & 0 & -0.05 \\ -1 & 0 & 1 & 0 \\ 0 & -0.05 & 0 & 0.05 \end{bmatrix}$$

$$R = 0.01$$

$$K = \begin{bmatrix} 54.725 & 3.1799 & -34.725 & -1.5447 \end{bmatrix}$$

68. Now simulate and get plots for voltage and all of the states.

69. Compare the response of all the states of this controller to the partial state feedback controller. What are the differences? What are the similarities?

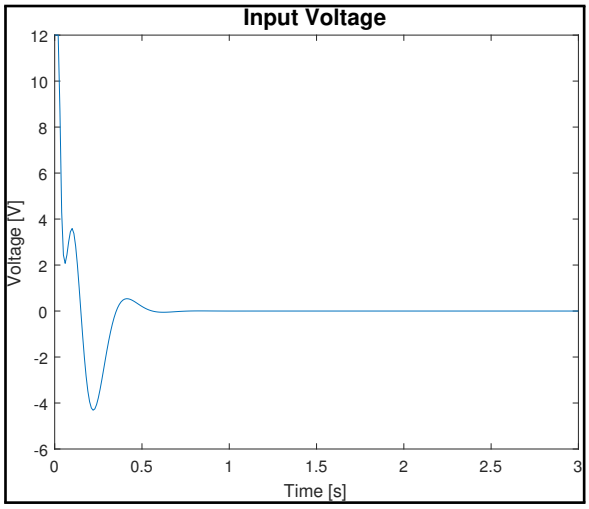



Figure 9.51: Full State Feedback Simulated Input Voltage

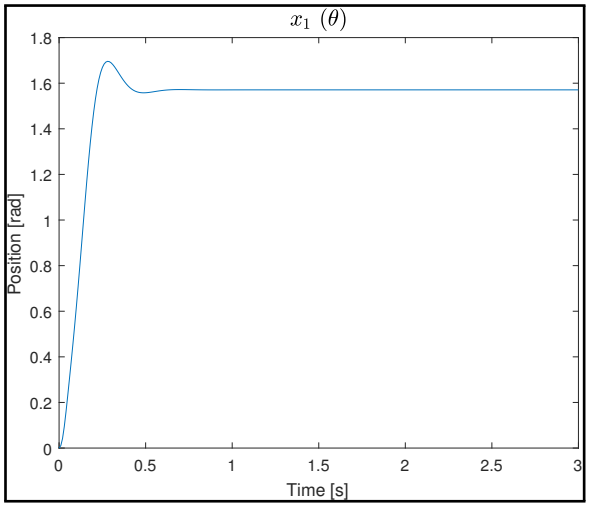



Figure 9.52: Full State Feedback Simulated Load Position

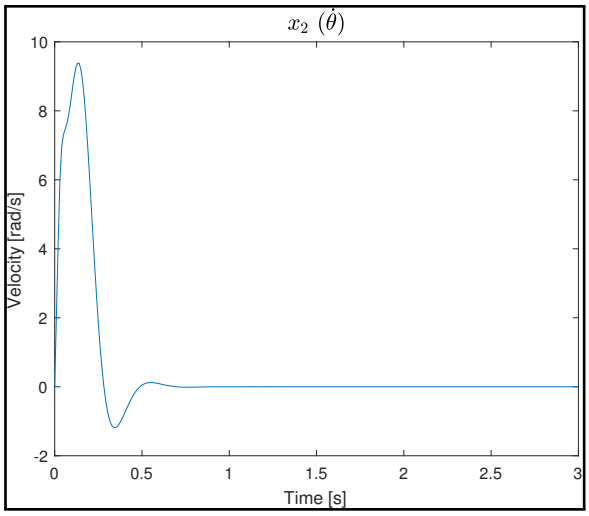



Figure 9.53: Full State Feedback Simulated Load Velocity

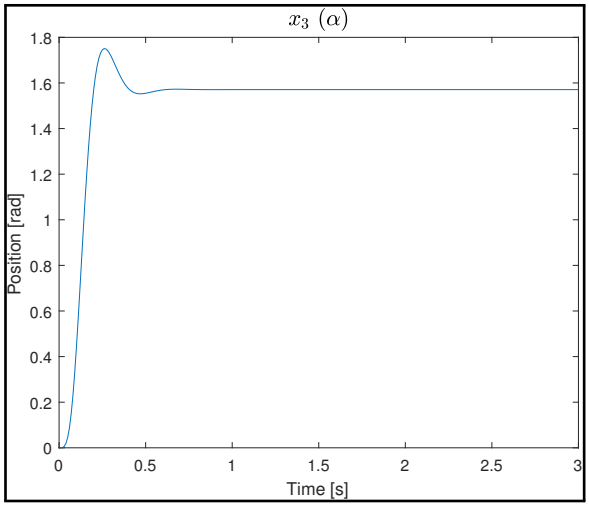



Figure 9.54: Full State Feedback Simulated Load Plus Link Deflection Position

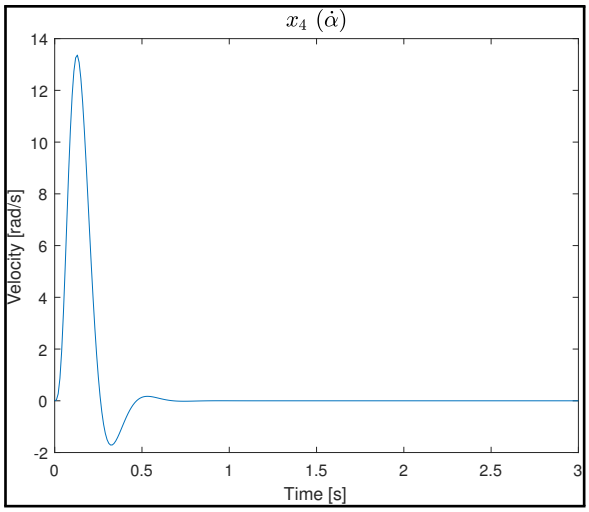



Figure 9.55: Full State Feedback Simulated Load Plus Link Deflection Velocity

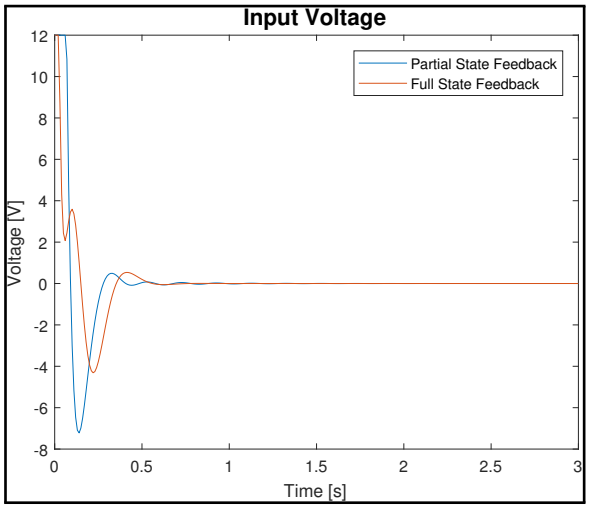



Figure 9.56: Partial and Full State Feedback Simulation Voltage

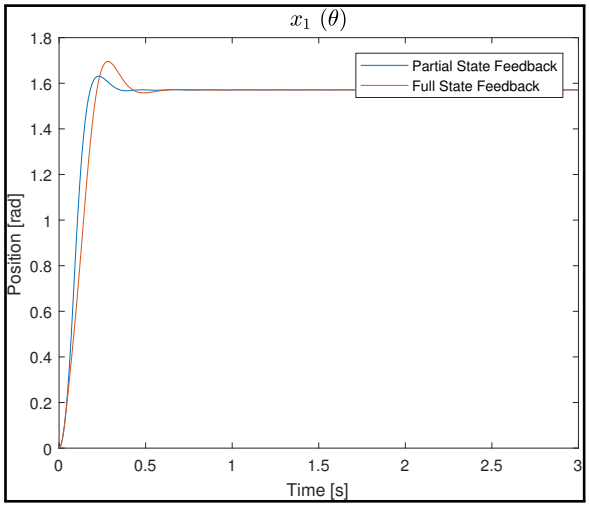



Figure 9.57: Partial and Full State Feedback Simulation $x_1$

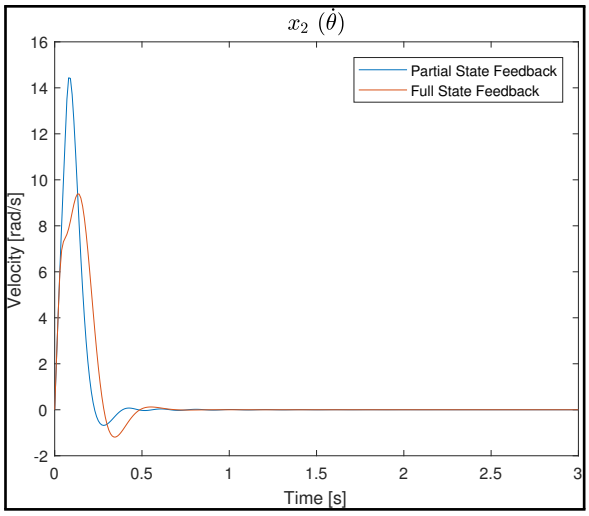



Figure 9.58: Partial and Full State Feedback Simulation $x_2$

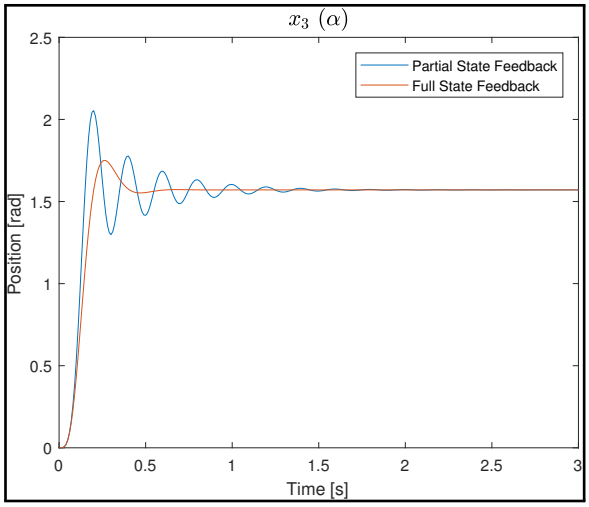



Figure 9.59: Partial and Full State Feedback Simulation $x_3$

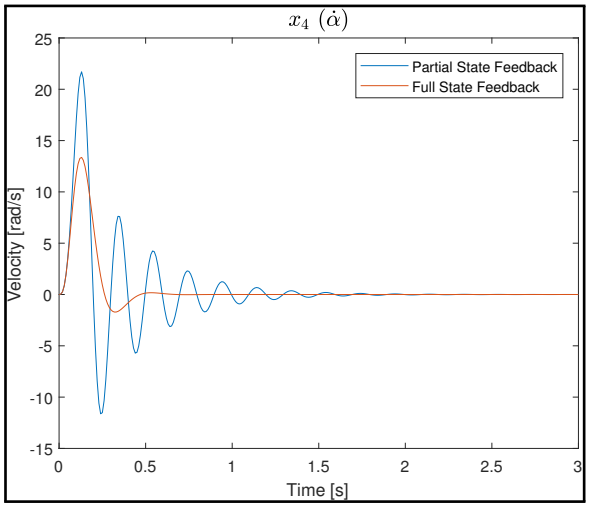



Figure 9.60: Partial and Full State Feedback Simulation $x_4$

- The most important thing to note is that the beam deflection in the full state feedback is minimized and does not oscillate around the reference point

---

### 9.5.7 Exercise 13: Experiment - Full State Feedback Optimal Control

70. To experimentally collect data for the full state feedback controller, change the switching constant block to -1 in the **RFL_exp.slx** Simulink file.

71. Run the experiment and compare with simulations.

72. How similar are your experiments to your simulations?

73. Describe the differences in link deflections between the two controllers.

74. Experiment with different control gains by adjusting Q and R and discuss any findings you may have.

---

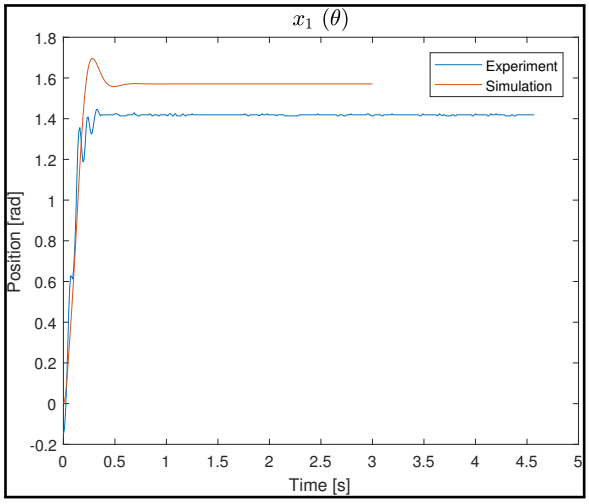



Figure 9.61: Full State Feedback Experimental and Simulated Load Position

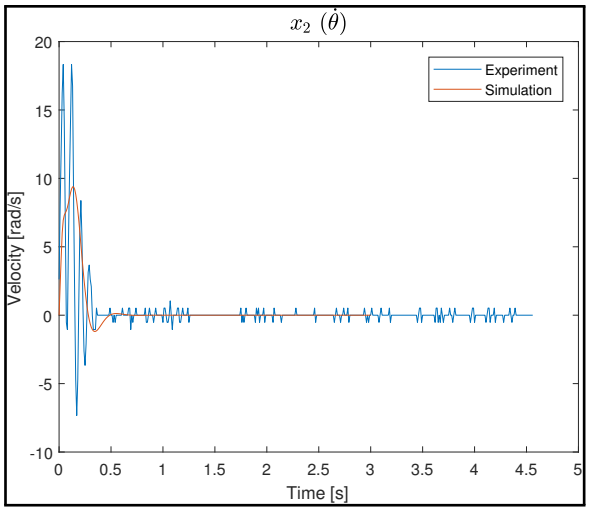



Figure 9.62: Full State Feedback Experimental and Simulated Load Velocity

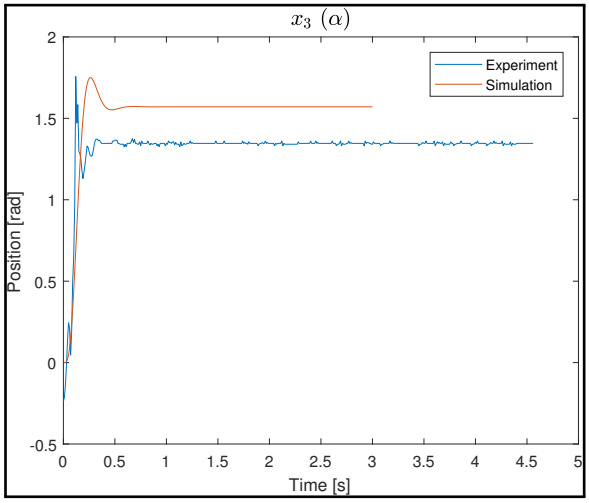



Figure 9.63: Full State Feedback Experimental and Simulated Load Plus Link Deflection Position

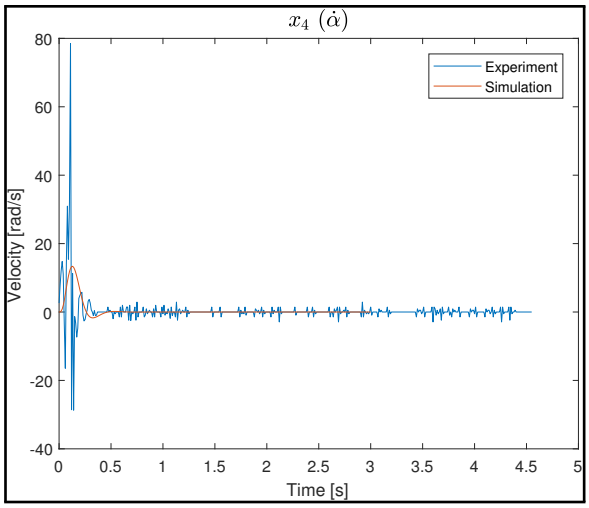



Figure 9.64: Full State Feedback Experimental and Simulated Load Plus Link Deflection Velocity

- Although the simulations and experiments do not exactly match, approximate modeling of the link shows that the beam deflection approaches the desired position and reaches zero velocity much quicker with the full state feedback versus the partial state feedback in both simulation and experiment.

---

## 9.6 Conclusion

This experiment consisted of designing two optimal stated feedback controllers for the rotary flexible link system: partial state and full state. A motor model was derived by experimentally obtaining the open loop step response. Then the dynamics of the link were derived by perturbing the link. A controller was designed and simulated based on these dynamics. Then the controller was tested on the physical system to make comparisons to the simulations.

# CHAPTER 10

# CONCLUSION

This thesis has discussed the concept of Take Home Labs. Take Home Labs is a series of inexpensive labs that can be done at home. These high-quality labs give students a better opportunity to apply important theoretical concepts learned in engineering courses that universities do not typically offer. The experiment handouts, 3-D printing files, and information on obtaining materials can found on the THL website, making the labs easily accessible (thl.okstate.edu/). The use of MATLAB and Simulink as the software for the labs gives students the chance to get experience with powerful and widely used programs. This thesis also discussed issues with the labs when the experiments were first performed by students in 2015, the changes made to the labs for 2017, and the assessment of the labs using Piazza. Some of the issues observed in 2015 were the real-time plotting function (serial plot), the labs taking too long, and students skipping important steps.

The serial plot function was sometimes difficult to use and could be frustrating for some students. Once this became an issue, the students appeared to be putting in less effort, suggesting that the serial plot was one of the root causes of the lack of student engagement. Fixing the serial plot function was one of the necessary changes. Upon assessing the labs in 2017, none of the students had any issues with getting the updated serial plot to work.

In 2015, a fair amount of students complained about the labs taking too long. In or-

der to increase the amount of student engagement, the number of labs performed was decreased from seven to five. We also shortened some of the labs by removing material that distracted the students from the main objective. It was observed that in 2017 there were not any complaints from the students that the labs took too long.

The overall idea behind the Take Home labs is to encourage students to make comparisons among theory, simulation, and experiment. The reports from 2015 showed that the majority of students skipped crucial discussions and questions. To ensure that a step is not skipped, stronger emphasis on proper comparison among theory, simulation, and experiment was made in the experiment handouts. Also, we put a summary table at the end of every handout with a list of every question or request for discussions that appeared throughout the lab. The quality of student reports increased significantly in 2017 versus 2015.

Another very important addition to THL was the incorporation of the web-based Social Question Answering platform Piazza. Using this platform, students connected with one another, as well as with the professor, which gave students an in-lab feel. Because they were able to get questions quickly answered at almost any time of the day, it almost certainly contributed to students being able to complete the labs more quickly. Using Piazza to assess the Take Home Labs was also important. We were able to capture statistics on any survey question posed, get feedback via student activity, and provide the students with answers to any questions they had. Any problems that came up in the labs was quickly addressed before they could become serious issues.

Based on all of the assessment that was performed as part of this research, the changes that were made to THL since the 2015 Systems Dynamics course were very successful. There was a vast increase in the number of students turning their lab reports in on time

in 2017, when compared with 2015. Student activity on Piazza was steady throughout the semester and did not show the last minute rush that happened in 2015. The pre-lab and post-lab survey questions generally indicated that student comprehension of key concepts was strengthened by performing the experiments. Finally, the lab reports in 2017 were more complete and contained more discussion and thoughtful analysis than those in 2015.

## 10.1 Future Work

Since Take Home Labs is very adaptable, updates can always be made to improve the labs. Adding more labs to THL can be a future improvement. Currently, Arduino is the only microcontroller used in THL. MATLAB and Simulink also have the capabilities for Raspberry Pi. Future improvements could include further investigation into the use of different microcontrollers. The experiment handouts were all created with a PC. MAC OSx support is another future improvement. On a MAC, some of the windows and other things in MATLAB and Simulink look a little different, which could be confusing for some students. The same experiment handout could be written, but all of the images were taken on a MAC. A frequently asked questions (FAQ) page could also be a helpful addition to THL. That way if students get stuck on a certain part, maybe they can find an answer on the FAQ page.

**BIBLIOGRAPHY**


[1] C. Pelton, "Take home labs and the ball and beam," Master's thesis, Oklahoma State University, Stillwater, OK, 2015.

[2] S. Hendrix, "Take home labs and the furuta pendulum," Master's thesis, Oklahoma State University, Stillwater, OK, 2015.

[3] W. Durfee, P. Li, and D. Waletzko, "At-home system and controls laboratories," *In Proceedings of the American Society of Engineering Education Annual Conference and Exposition*, 2005.

[4] J. P. Oliver and F. Haim, "Lab at home: Hardware kits for a digital design lab," *Education, IEEE Transactions*, vol. 52, no. 1, pp. 46–51, 2009.

[5] M. Jouaneh and W. J. Palm., "Control systems take-home experiments [focus on education]," *IEEE Control Systems*, vol. 33, no. 4, pp. 44–53, 2013.

[6] J. Sarik and I. Kymissis, "Lab kits using the arduino prototyping platform," *In Frontiers in Education Conference (FIE)*, pp. T3C–1, 2010.

[7] Y. S. Lee, Y. S. Park, B. E. Jo, and S. Seo, "Low-cost rcp system for control courses using matlab and the open-source hardware," *In World Congress on Mechanical, Chemical, and Material Education (MCM 2015)*, 2015.

[8] J. Crist, J. Littlefield, K. Kim, P. Kruse, B. Sohn, K. Strauss, and W. Durfee, "Me3281 take-home lab kit me4054w senior design," *tech. rep., University of Minnesota*, 2013.

[9] J. Yao, L. Limberis, and S. Warren, "Work in progress - a ubiquitous laboratory model to enhance learning in electronics courses offered by two universities with dissimilar curricula," pp. F3C–1, 2010.

[10] S. Warren, X. Dong, and J. Yao, "A rapid analysis and signal conditioning laboratory (rascl) design compatible with the national instruments mydaq® platform," *In Annual Conference and Exposition, American Society for Engineering Education*, 2011.

[11] "Quanser." `http://www.quanser.com/`. Accessed: 2018-03-22.

[12] "The miniature balacing robot: A low-cost mobile lab experiment kit for education." `https://minseg.com/`. Accessed: 2018-03-22.

[13] M. T. Hagan, C. A. Hernandez, W.-C. Yeung, and M. Hozhabri, "A control systems lboratory with microcomputer supervision," *IEEE Control Systems Magazine*, vol. 4, pp. 15–19, 1984.

[14] M. T. Hagan and C. D. Latino, "A modular control systems laboratory," *Computer Applications in Engineering Education*, vol. 3, no. 2, pp. 89–96, 1995.

[15] A. Chevalier, C. Copot, C. Ionescu, and R. D. Keyser, "A three-year feedback study of a remote laboratory used in control engineering studies," *IEEE Transactions On Education*, vol. 60, no. 2, pp. 127–133, 2017.

[16] M. J. Blooma, J. C. Kurian, A. Y. K. Chua, D. H. L. Goh, and N. H. Lien, "Social question answering: Analyzing knowledge, cognitive processes and social dimensions of micro-collaboration," *Computers and Education*, vol. 69, pp. 109–120, 2013.

[17] J. Blooma, "Micro-collaborations in piazza," *Proceedings of the SIGED: IAIM Conference*, 2013.

[18] A. Hughes, *Electric Motors and Drives Fundamentals, Types and Applications.* Elsevier Ltd., 3rd ed., 2006.

# APPENDIX A

## rc script.m

```
close all; clear all; clc;

%Slow period
T = 6;

figure(1)
sim('rc')
plot(tout,in)
hold on
plot(tout,out,'r')
xlabel('time [s]')
ylabel('amplitude')
legend('input','output')
title('T = 6')
hold off

%Faster period
T = 0.5;

figure(2)
sim('rc')
plot(tout,in)
hold on
plot(tout,out,'r')
xlabel('time [s]')
ylabel('amplitude')
legend('input','output')
title('T=0.5')
hold off

%close to arduino period
T = 1/500;
```

```
figure(3)
sim('rc')
plot(tout,in)
hold on
plot(tout,out,'r')
xlabel('time [s]')
ylabel('amplitude')
legend('input','output')
title('T=0.002')
hold off
```

# APPENDIX B

## FindShift2.m

```
function [y_pulse,min_rmse,ref,start] = findShift2(y,period,pulsewidth,magnitude)


y = y(:)';
ltot = length(y);


periods = ceil(ltot/period);
ref_base = [magnitude*ones(1,pulsewidth) zeros(1,period-pulsewidth)];
ref_tot = [];
for i=1:periods,
    ref_tot = [ref_tot ref_base];
end
min_sse = inf;
sse = zeros(1,period);
for i=1:period,
    ref = circshift(ref_tot,[0 i-1]);
    ref = ref(1:ltot);
    sse(i) = sum((y-ref).^2);
end
[~,start]=min(sse);


%offset = ceil(.1*pulsewidth);
offset = min(ceil(.1*pulsewidth),(start-1));
ytemp = y(start-offset:start);
dytemp = diff(ytemp);
offset_ind = find(dytemp>0,1,'first');
if ~isempty(offset_ind)
    start = start-offset+offset_ind-1;
end
%start = start-offset+offset_ind-1;


dy = diff(y);
dy1 = dy(1:(start+1));
```

```
dy1 = dy1(:)';
ind = find(fliplr(dy1)==0);
if ~isempty(ind),
    start = start - ind(1);
else
    dy1 = dy((period+1):(period+start+1));
    dy1 = dy1(:)';
    ind = find(fliplr(dy1)==0);
    if ~isempty(ind),
        start = period + start - ind(1);
    else
        start = 1;
    end
end

ref = circshift(ref_tot,[0 start-1]);
ref = ref(1:ltot);
sse = sum((y-ref).^2);
y_pulse = y(start:(start+pulsewidth-1));

min_rmse = sqrt(sse/ltot);
```

## APPENDIX C

### 2015 Project Guidelines

During the semester, you will be performing several laboratory experiments that will reinforce/enhance theoretical concepts that are covered throughout the course. In addition, you will design and implement a feedback control system for a DC motor as a final project. Together, these experiments and the final project will make up 25% of your course grade, as described in the syllabus. The experiments can all be done at home, at your own pace.

As you work through each experiment (and the final project), you will create an electronic document (lab notebook), where you will record the work that you perform. This will include theoretical calculations (e.g., partial fraction expansions, block diagram reductions, etc.), plots created in MATLAB/Simulink from simulations and experiments, discussions of results, etc. As with quizzes and exams, your work and your discussions are more important than the numerical "answers" that you report. (In fact, there is no single correct answer for most of the experiments or the final project. The answers will only be considered "correct" if you can justify them with your work and your discussion.)

After you complete each experiment, you will submit a PDF version of your lab notebook. It should contain the results of that experiment, as well as all the experiments you have performed to that point. The notebook will be checked for completeness, and comments may be provided to assist you in improving your notebook. You can go back and make changes to previous experiment write-ups until the final notebook (including the design project) is turned in at the end of the semester. A grade will be assigned to the notebook only at the end of the semester.

You can perform the experiments in teams of two. Notify me at your earliest convenience of the members of your team. Although you can collect data together, and discuss the operation of the experiment, each person on the team should do their own work, and write up their own discussion. In some cases, the experimental setup will be slightly different for the two members (e.g., different loads for the motor). Each team member will turn in his or her own lab notebook.

The experiments can be accessed through the Take Home Labs website (thl.okstate.edu). If you click on the Experiments -> All links, you will arrive at a page that lists all of the experiment names. Click on the name of the experiment that you are going to perform. On the resulting page, you will find an experiment handout that will guide you through all of the steps of the experiment. There will also be links to parts that you may need to order to complete the experiment, and software that you can download to assist in the operation of the experiment.

The experiments you should perform, and their due dates, are listed below.

| Experiment Title | Due Date |
|---|---|
| Blinking LED | September 11, 2015 |
| Simple DC Motor | September 25, 2015 |
| Sampling and Data Acquisition | October 9, 2015 |
| Open Loop Step Response | October 23, 2015 |
| Open Loop Frequency Response | November 6, 2015 |
| Closed Loop Step Response | November 20, 2015 |
| Root Locus Control Design | December 4, 2015 |

## APPENDIX D

### 2017 Project Guidlines

During the semester, you will be performing a design project that will reinforce/enhance theoretical concepts that are covered throughout the course. At the end of the project, you will design and implement a feedback control system for a DC motor. The final project will make up 25% of your course grade, as described in the syllabus. The project can be done at home, at your own pace.

There will be a set of five experiments that you will perform as part of the project. As you work through each experiment, you will create an electronic document (project notebook), where you will record the work that you perform. This will include theoretical calculations (e.g., partial fraction expansions, block diagram reductions, etc.), plots created in MATLAB/Simulink from simulations and experiments, discussions of results, etc. As with quizzes and exams, your work and your discussions are more important than the numerical "answers" that you report. (In fact, there is no single correct answer for most of the experiments or the final control design. The answers will only be considered "correct" if you can justify them with your work and your discussion.)

After you complete each experiment, you will submit a PDF version of your project notebook. It should contain the results of that experiment, as well as all the experiments you have performed to that point. (The full notebook is just a continuation of each experiment, and you just need to keep adding to the notebook as you go.) The notebook will be checked for completeness, and comments may be provided to assist you in improving your notebook. You can go back and make changes to previous experiment write-ups until the final notebook is turned in at the end of the semester. A grade will be

assigned to the notebook only at the end of the semester.

You can perform the project in teams of two. Although you can collect data together, and discuss the operation of the experiments, each person on the team should do their own work, and write up their own discussion. In some cases, the experimental setup will be slightly different for the two members (e.g., different loads for the motor). Each team member will turn in his or her own project notebook.

The experiments can be accessed through the Take Home Labs website (thl.okstate.edu). If you click on the Experiments -> All links, you will arrive at a page that lists all of the experiment names. Click on the name of the experiment that you are going to perform. On the resulting page, you will find an experiment handout that will guide you through all of the steps of the experiment. There will also be links to software that you can download to assist in the operation of the experiment.

Piazza will be used to make it easier to interact with the professor, TA, and other students. Whenever you have questions on any of the experiments, please post your question to Piazza. We will try to answer each question as soon as we can, and you may find that other students can help also. We will also post some survey questions before and after each experiment to help us understand how the project is progressing. There are no wrong answers to these questions, but if you don′t answer the questions, it can affect your final project grade.

The experiments you should perform as part of the project, and their due dates, are listed below.

| Experiment Title | Due Date |
|---|---|
| Simple DC Motor | October 6, 2017 |
| Sampling and Data Acquisition | October 20, 2017 |
| Open Loop Step Response | November 3, 2017 |
| Closed Loop Step Response | November 17, 2017 |
| Root Locus Control Design | December 8, 2017 |

**Project Notebook Notes**

- On each due date, the full notebook should be submitted to the D2L/Brightspace dropbox as a PDF file. Only PDF files will be accepted. You will not be given a grade until the final notebook is turned in, but if you do not turn in the notebook at each due date, it can affect your final grade.

- Each experiment should form one section of the notebook. The front page of the notebook should have your name and the name of your project partner.

- Include all figures that you generate while performing the experiment, including those generated by Matlab as well as hand-drawn figures.

- Include all theoretical calculations.

- In the experiment handouts, there is a number for each step of the experiment. Use these same step numbers to identify the parts of your report.

- The most important parts of your report are your answers to questions and your discussions of the results and what they mean. Your answers and discussions should be extensive and should demonstrate your engagement in the experiment.

- There will be a number of questions asked in each the experiment. When you answer these questions, first write out the question, and then your answer.

- An important part of most experiments will be a study of the differences among theory, simulation and experimental results. In your report, when you find differences between any of these three components, be sure to carefully explain why you think the differences occur.

- It is important that you turn in your project notebook on each of the due dates listed above. We will review your work and provide suggestions for improvement, if needed, each time. After you receive the suggestions, you can change any parts

of your notebook, even for previous experiments. The notebook will only receive a grade at the end of the semester, when you turn in the final version of the notebook on December 8.

# APPENDIX E

## Simple DC Motor Handout

### E.1 Objective

This lab serves as the "Hello World" lab, common to most computer science courses. The objective is to introduce the basic hardware and software components that will be used in later experiments.

### E.2 Setup

This section of the lab handout describes all of the required materials needed to make the lab work correctly. There are two subsections, hardware and software. The hardware subsection lists the physical materials you will need, and describes how they are physically connected. The software subsection describes how to download the software needed to run the lab.

#### E.2.1 Required Materials

This section lists all of the software and hardware materials that you will need to have available before performing this experiment. It also lists any prior experiments that should be performed before running this experiment.

**Hardware**

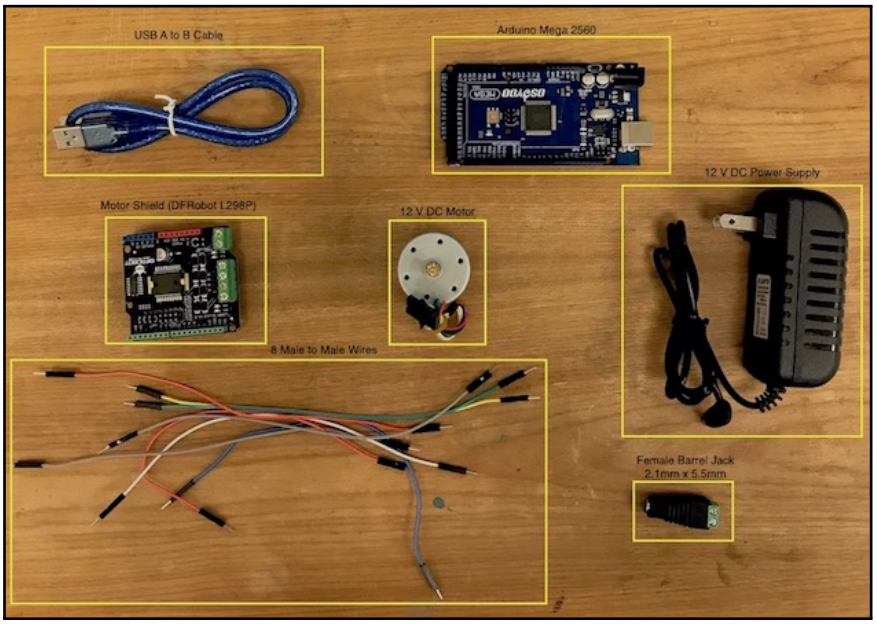



Figure E.1: Hardware Required for Laboratory

**Hardware**

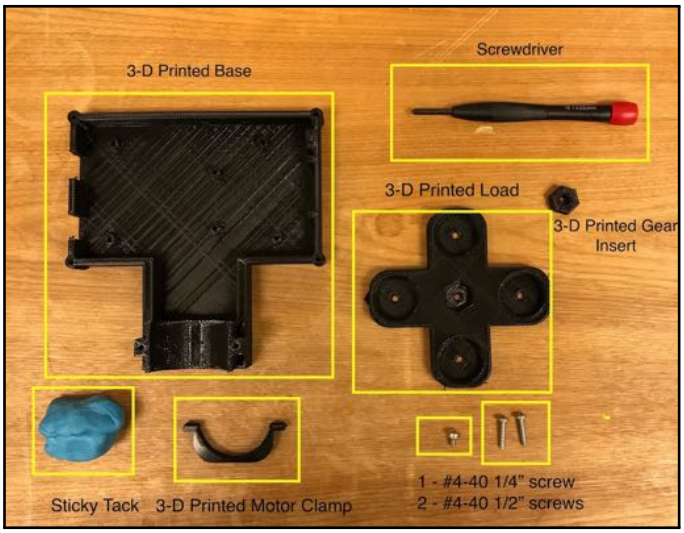



Figure E.2: Hardware Required for Laboratory

- DC Motor (Pololu item 2821 - Motor with 64 CPR Encoder no gearbox)
- Microcontroller (Arduino Mega 2560)

- Motor Shield (DFRobot L298P)
- 12 V DC Power Supply (Universal AC Adapter with 2.1mm x 5.5mm Male Connector)
- USB B to A Converter Cable (USB 2.0 A-Male to B-Male Cable)
- 8 Wires (20cm Male To Male Jumper Wire)
- Female Barrel Jack (2.1mm x 5.5mm Female CCTV Power Jack Adapter)
- 3-D Printed Base
- 3-D Printed Load
- 3-D Printed Motor Clamp
- 3-D Printed Load
- 3-D Printed Gear Insert
- Screwdriver
- Sticky Tack
- 1 - #4-40 1/4" screw
- 2 - #4-40 1/2" screws

**Software**

- Matlab/Simulink 2017a
- Windows 10 or Windows 7

**Prerequisite Experiments**

- This is an introductory experiment and does not require that any other experiments be performed first

### E.2.2 Software Setup

The necessary software files that are needed for this experiment can be downloaded from the Take Home Labs webpage. One method for downloading the files is shown in the steps below.

1. Open your internet browser and navigate to `thl.okstate.edu`
2. On the left side of the Homepage, select “Courses"
3. In the middle section of the Courses page select “System Dynamics"
4. In the middle section of the System Dynamics page find and select *Simple DC Motor*.
5. On the Simple DC Motor page select “Software/Code" in the rightmost section. A zipfile named *SoftwareSimpleDC* should download.
6. Right-click the file and choose “Extract All...", or any other method of extracting files on your PC.
7. Extract this folder somewhere convenient, and remember the location. This will be the folder where all of the files and plots created for this experiment are saved.

### E.2.3 Hardware Setup

This section describes the step-by-step hardware setup of the DC motor lab. The DC motor used in this lab will be utilized in later labs, and this lab will develop a familiarity with how the motor is interfaced to an Arduino using Simulink. A circuit diagram for the

DC motor will be included as well as a photo of the circuit connection.

### Configuring Motor Shield

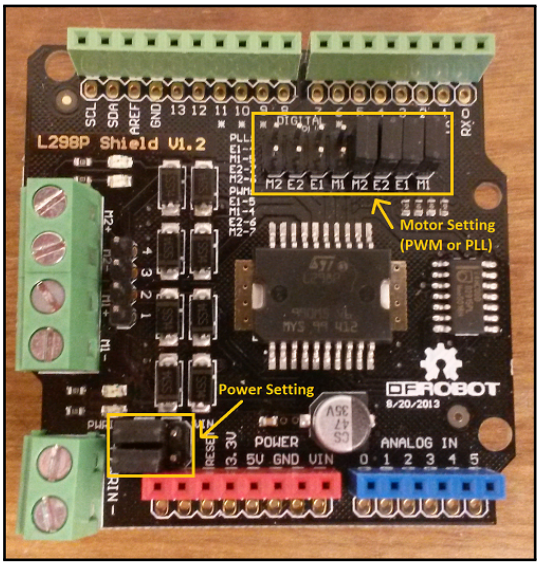



Figure E.3: Correct Motor Shield/ Arduino Connection

1. Jumper the pins of the DFRobot motor shield as a PWM input with external power (power outside of the Arduino Mega). Figure E.3 shows the correct setup of the motor shield with jumpers on the right four sets of pins (labeled as M2, E2, E1, and M1 from left to right). These jumpers ensure that the motor shield is configured to accept PWM signals to drive the DC motor.

2. Additionally, the motor shield has 6 pins in the bottom left-hand corner of the board, which are also shown in Figure E.3. Jumper the pins to accept an external power supply in order to power the DC Motor. This requires that the left four pins out of the six pins are jumpered, with the top-left two pins jumpered together and the bottom-left two pins jumpered together.

**Connecting Hardware**

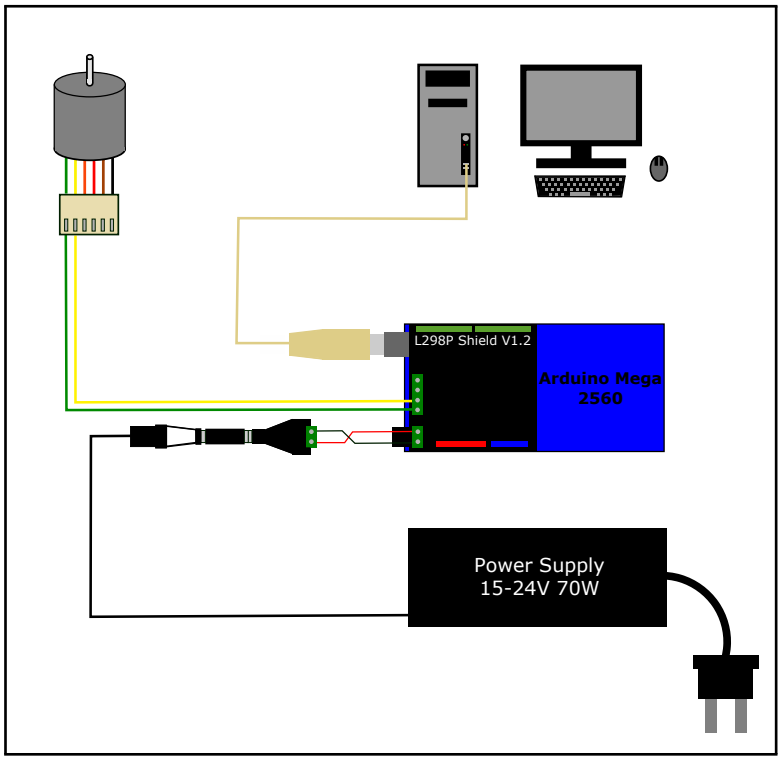



Figure E.4: Circuit Diagram for Simple DC Motor Hardware

3. Check to see that all hardware shown in Figures E.1 and E.2 is available.

4. Take the #4-40 1/4" screw and a screw the Arduino onto the 3-D printed base as shown in Figure E.5.

5. Take the motor shield and plug it into the Arduino board as seen in Figure E.6. Make sure that the pin labeled 5 on the motor shield is plugged into the A5 pin on the Arduino. Additionally, make sure that the Rx pin is aligned with the RX 0 pin on the Arduino. This will ensure that the motor shield is connected properly to the Arduino.

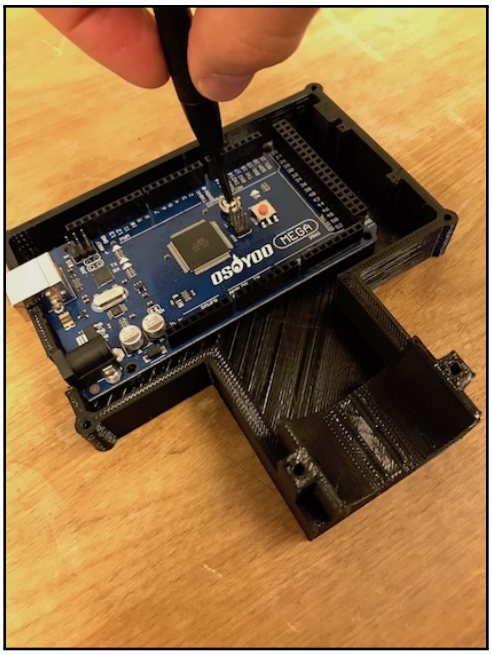


Figure E.5: Mounting Arduino Onto Base

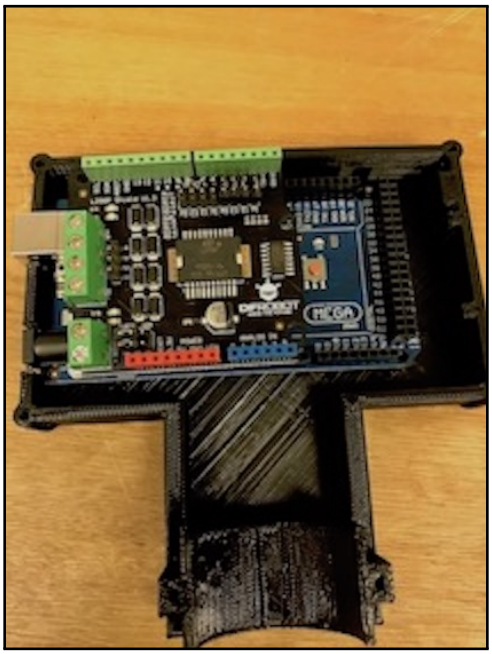


Figure E.6: Correct Motor Shield Configuration

6. Next, take the female barrel jack and screw on a red wire to the positive and a black wire to the negative end as shown in Figure E.7. Each connector port has screw terminals, so clamp them down on the wires by using a small flat-head screw driver.

7. Then, connect the other end to the motor shield labeled PWRIN where the positive wire goes to the positive power in and the negative wire goes to the negative power in as shown in Figure E.8.

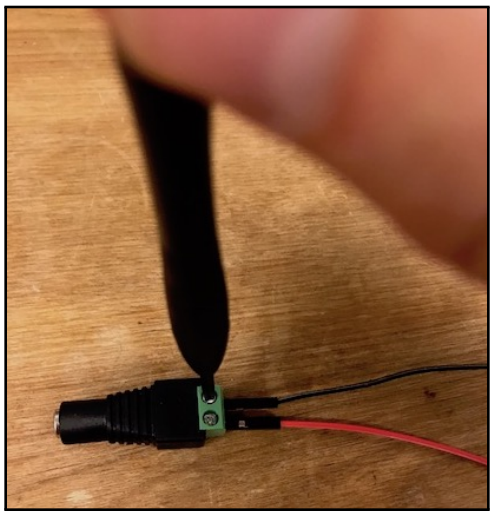


Figure E.7: Connecting Wires to the Barrel Jack

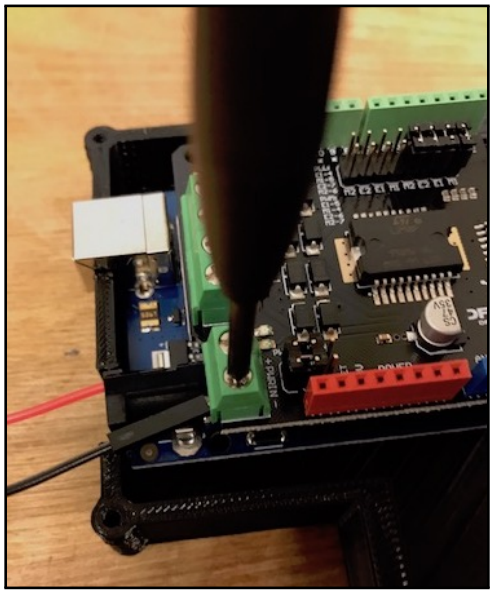


Figure E.8: Connecting Barrel Jack to Motor Shield

8. Now, take the DC motor and connect the other 6 wires into the motor wiring harness as shown in Figure E.9.

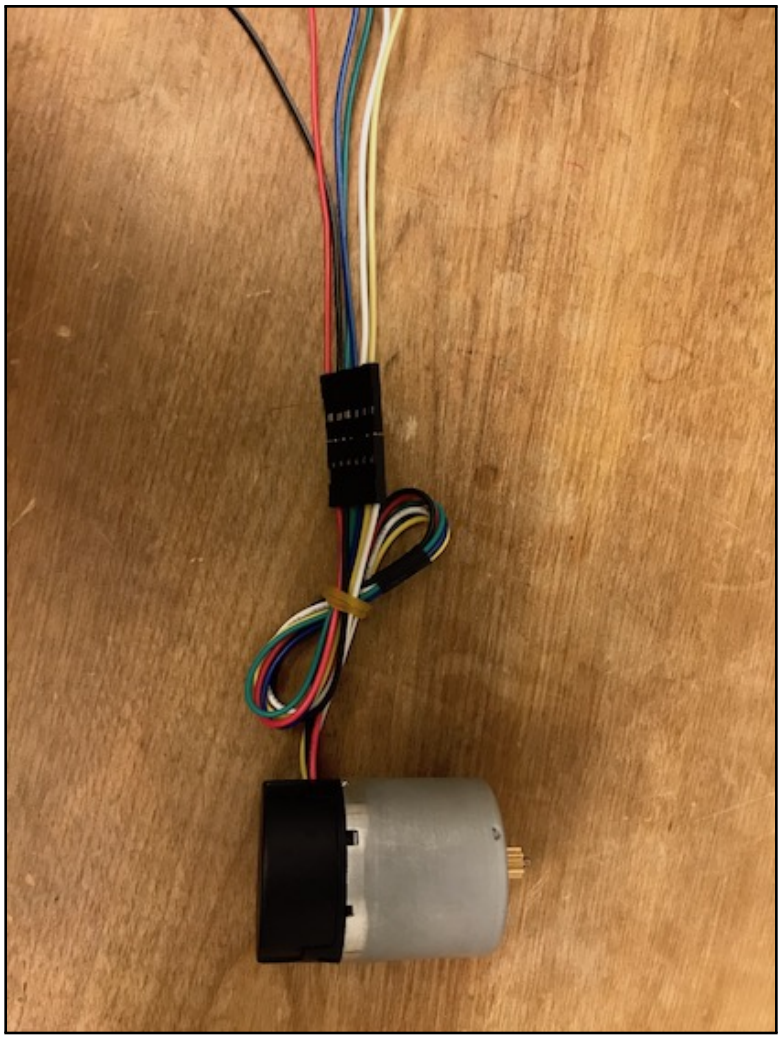


Figure E.9: Motor Wires

9. Next, mount the motor onto the 3-D printed base as shown in Figure E.10.

10. Then, take the 2 #4-40 1/2" screws and start threading the screws into the 3-D printed motor clamp as shown in Figure E.11.

11. Then, start screwing in the screws from the motor clamp into the base until the motor mount is tightly clamped down leaving a little bit of clearance where the motor is hanging over as shown in Figure E.12

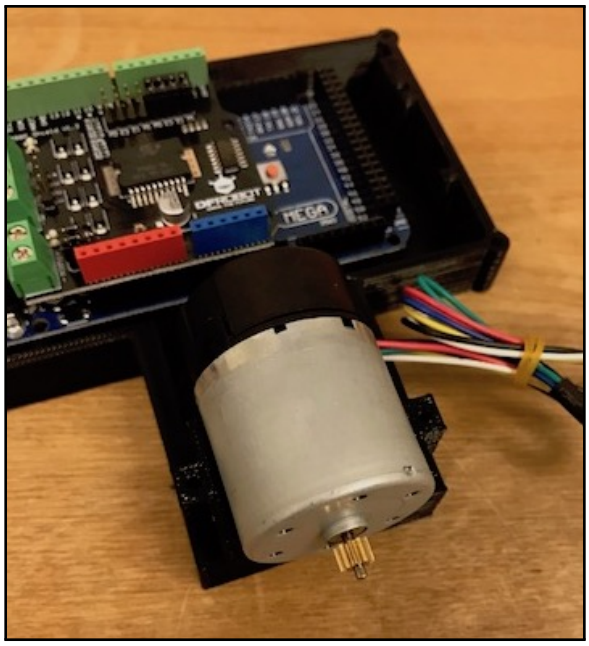


Figure E.10: Placing Motor on Base

Figure E.11: Threading Screws into Motor Clamp

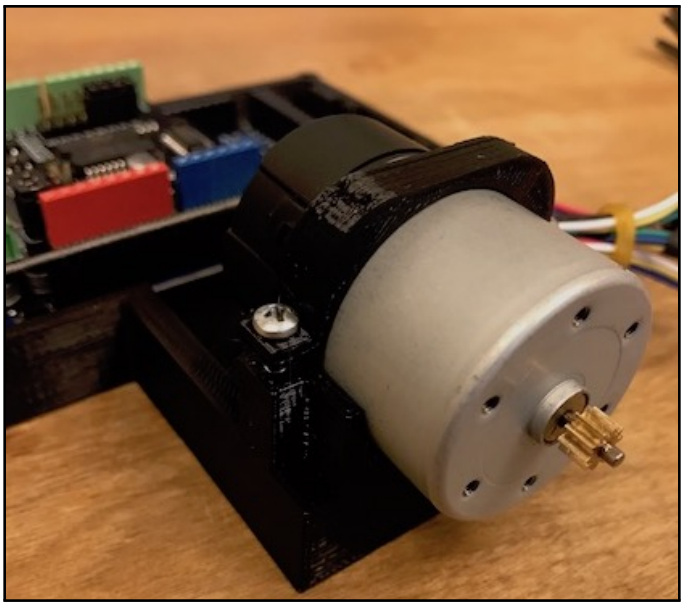


Figure E.12: Final Result of Mounting Motor

12. Now, take the red and black wires on the motor and connect them to M1+ and M1- on the motor shield respectively as shown in Figure E.13.

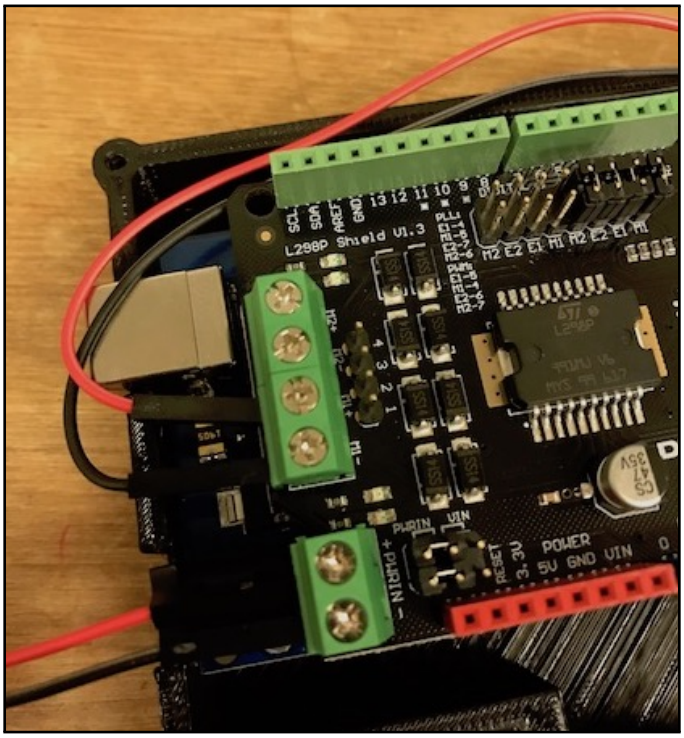



Figure E.13: Connecting Power to the Motor

13. Next, connect the motor encoder power and ground. Connect the green wire to the GND pin and the blue wire to the 5V pin as shown in Figure E.14.

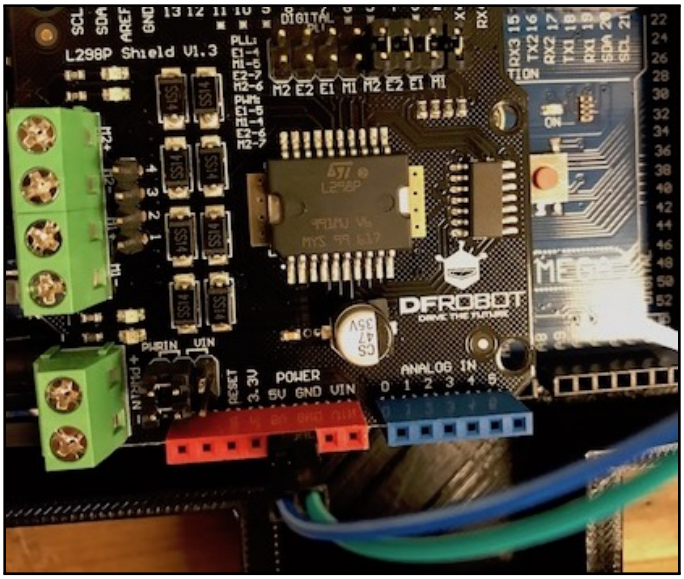



Figure E.14: Encoder Power

14. To finish up connecting the motor and encoder, connect the encoder wire A (yellow) to pin 2 on the motor shield and the encoder wire B (white) to pin 3 as shown in Figure E.15. Table E.1 gives a summary of how the wires need to be connected.

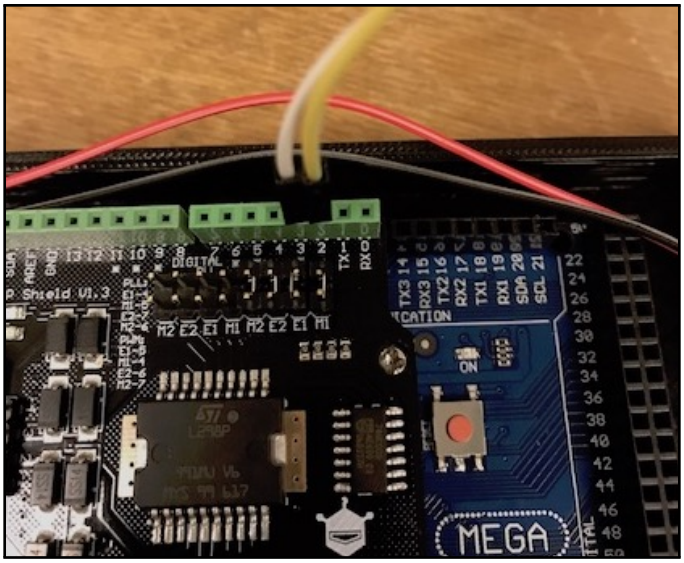



Figure E.15: Encoder Data Wires

Table E.1: Motor Wires

| Motor Wire | Motor Wire Color | Motor Shield Pin |
|---|---|---|
| Positive Power | Red | M1+ |
| Negative Power | Black | M1- |
| Encoder Power | Blue | 5V |
| Encoder Ground | Green | GND |
| A | Yellow | 2 |
| B | White | 3 |

15. In order to keep the base from being subject to vibration, sticky tack is used. Make six small spheres of sticky tack as shown in Figure E.16.

16. Then, place the spheres on the bottom side of the base as shown in Figure E.17.

17. Now, place the base on the edge of table, with the end slightly hanging off. It should be clear from anything that could interfere with the rotating load, as shown in Figure E.18.

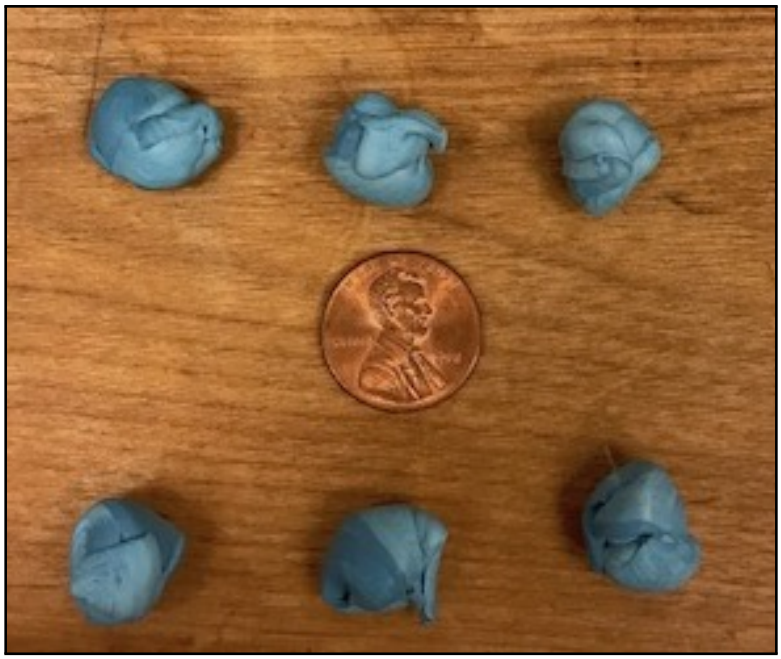


Figure E.16: Sticky Tack Spheres

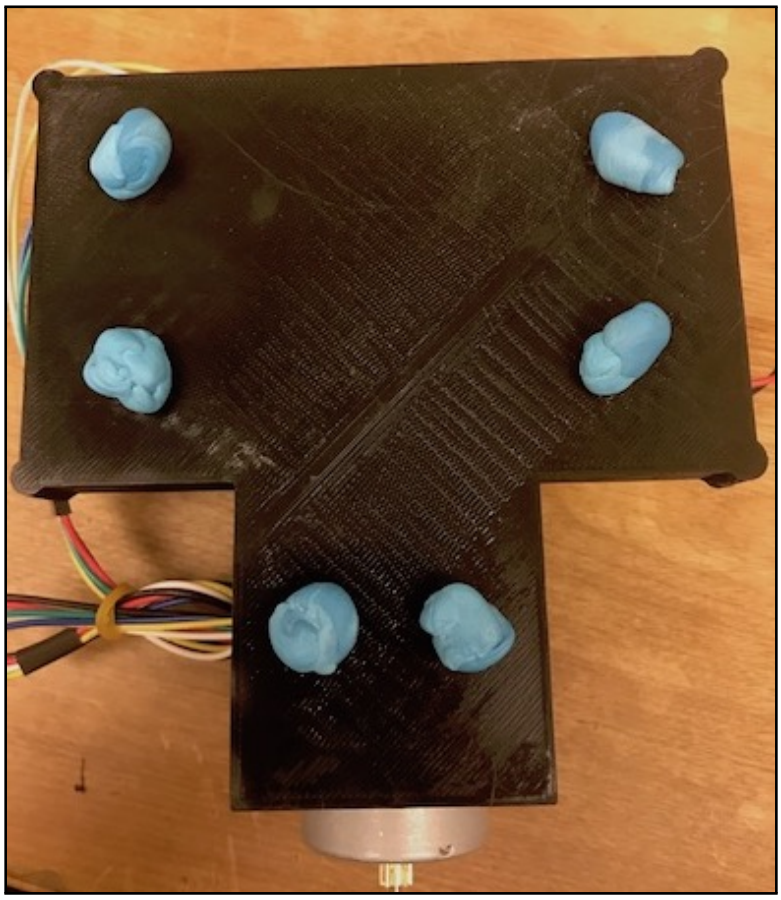


Figure E.17: Sticky Tack on Base

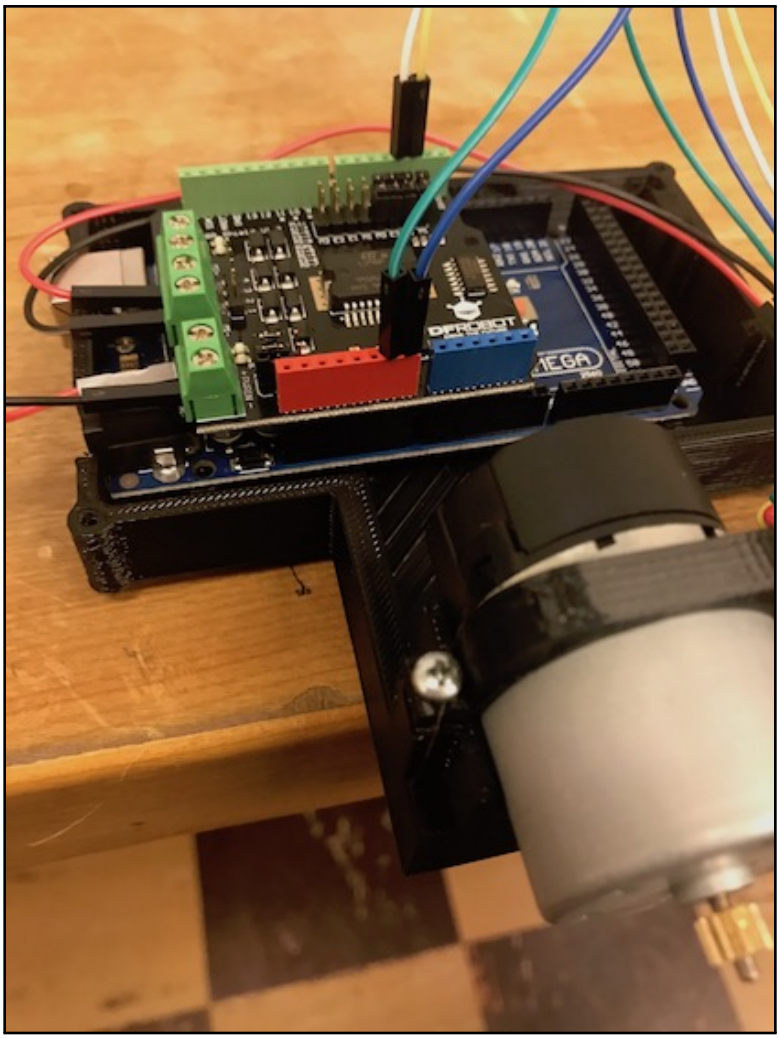



Figure E.18: Mounted Base

18. Next, insert the 3-D printed gear insert into the 3-D printed load as shown in Figure E.19. You may need to apply some force to get the insert in the load as it is a very snug fit.

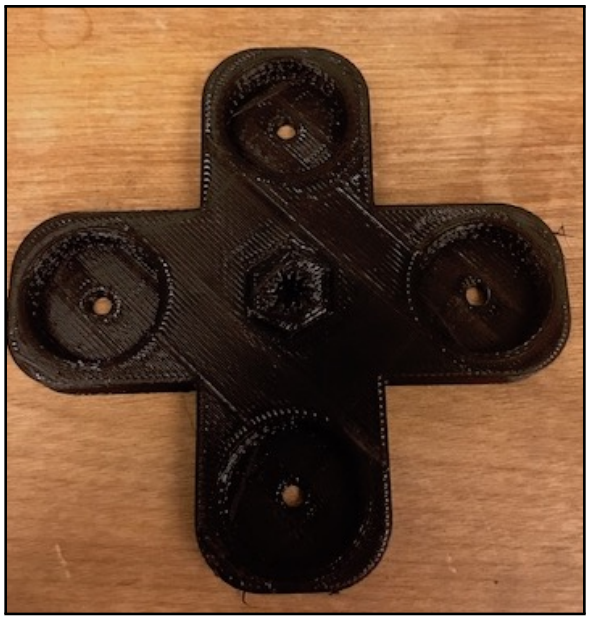


Figure E.19: Gear Insert in Load

19. Create an even smaller sphere of sticky tack shown in Figure E.20.

20. Then, put the small sphere of sticky tack into the gear insert as shown in Figure E.21.

21. Push the load onto the motor. The gear should slide into the gear insert. Make sure that the fit is snug and that the sticky tack holds on the load as shown in Figure E.22.

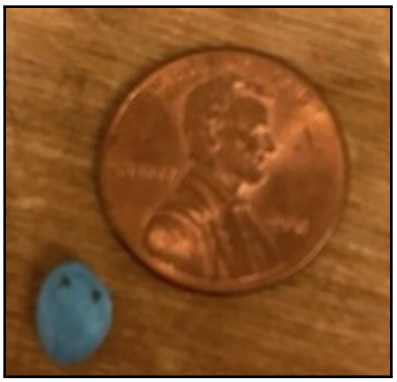


Figure E.20: Small Sphere of Sticky Tack

Figure E.21: Sticky Tack in Load

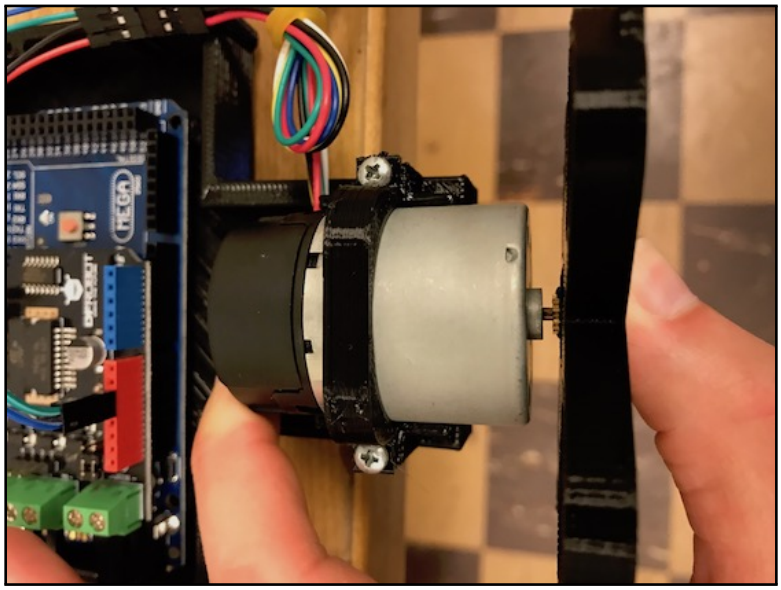


Figure E.22: Load On Motor

### E.2.4 Software Setup

This section will describe how to set up the Simulink diagram used to run the simple DC motor, as well as how to setup the drivers for the Arduino. Matlab 2017a with Simulink should be installed before beginning this section. (Other versions of Matlab may work, but the 2017a version has been tested.)

#### Installing Arduino Software

These steps are based on the installation method provided on the Arduino website. General information about getting started with the Arduino can be found at www.arduino.cc/en/Guide/H This is a good place to explore if you experience any difficulties with the following installation process.

22. In your web browser, navigate to www.arduino.cc.

23. At the top of the home page, select *Software*.


Figure E.23: Arduino Homepage

24. Then click on the windows installer link

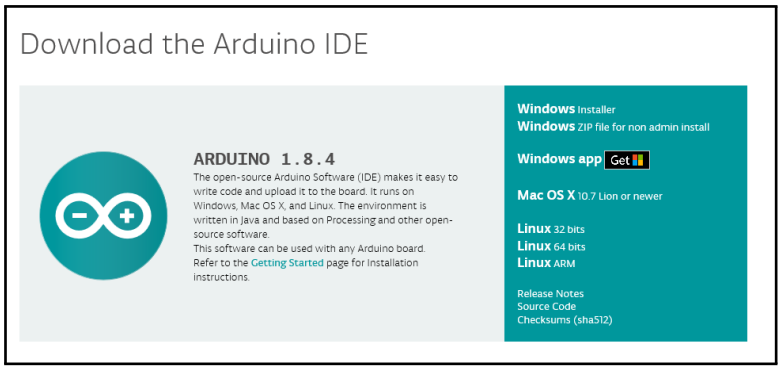



Figure E.24: Windows Installer

25. A page asking you to contribute should show, select J*ust Download*, unless you would like to contribute to the Arduino software.

26. An executable file will start to download. Once the download finishes, select the file and install the Arduino IDE. You will need administrator access on your PC to install the IDE.

27. When the install finishes, connect the USB cable from the Arduino to the computer. The green LED on the Arduino should turn on, indicating the board is being powered.

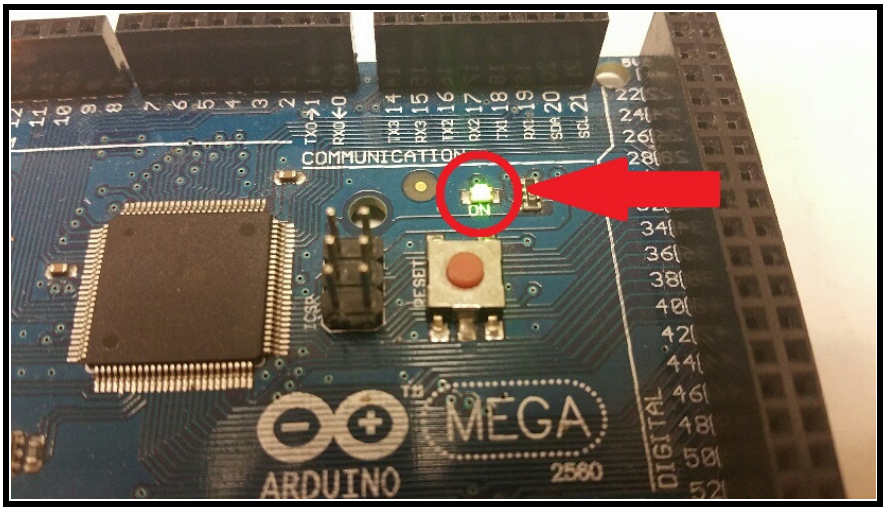



Figure E.25: Arduino Power ON LED

28. Wait for the computer to search for the drivers online.

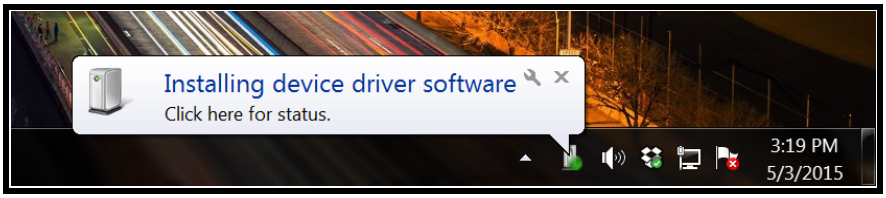



Figure E.26: Driver Search

29. Navigate to the Device Manager on the computer, and check that the Arduino shows under “Ports(COM & LPT)". The Arduino Mega should be listed as “Arduino

Mega 2560 (COMx)" where x is the communication port number being used. Write down the communication port number as it may be useful later on.

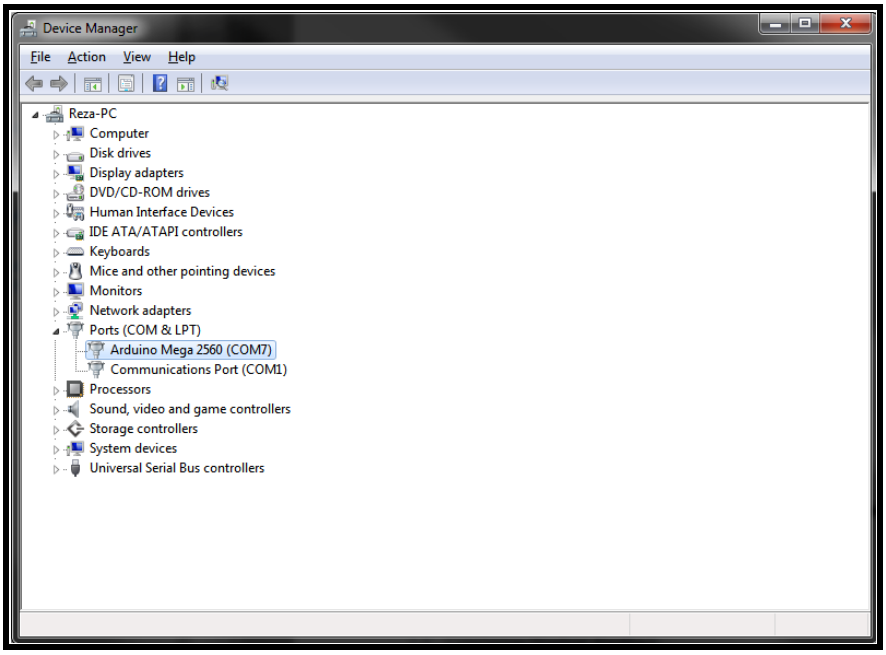



Figure E.27: Driver Check

### Installing Arduino Simulink

30. Open Matlab 2017a and type "supportPackageInstaller" into the command window.

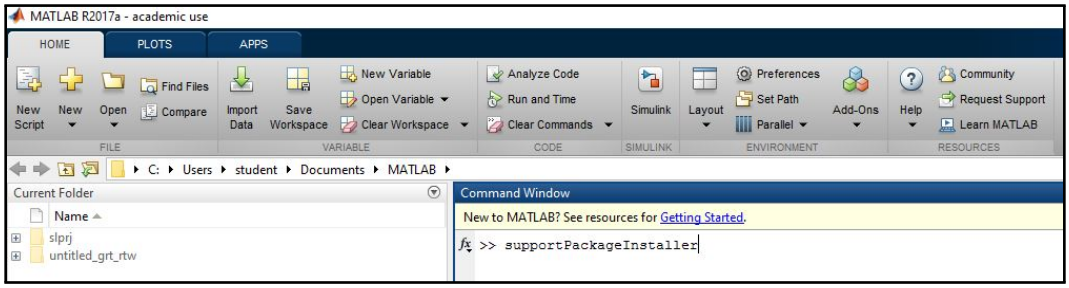



Figure E.28: supportPackageInstaller in Command Window

31. In the support package installer search bar type "Simulink Support Package for Arduino Hardware".

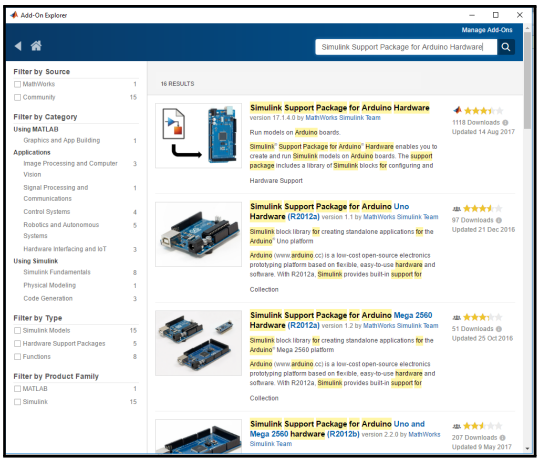



Figure E.29: Simulink Support Packages Search


Figure E.30: Simulink Support Package for Arduino Hardware

32. Click on *Simulink Support Package for Arduino Software* as shown in Figure E.29.

33. Click Install as shown in Figure E.30. **Note:** You may be prompted to log into with the Mathworks account tied to the license somewhere along this process. Logging in is necessary in order to download the support package.

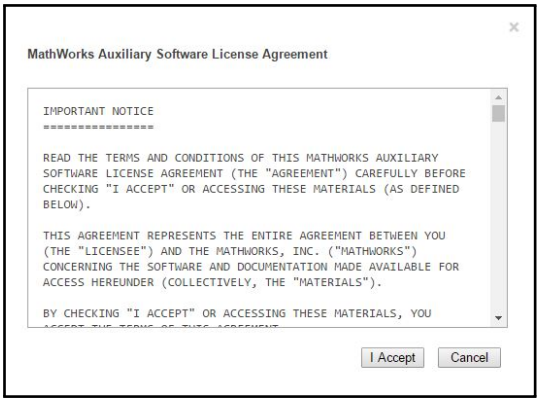



Figure E.31: Accepting Installation

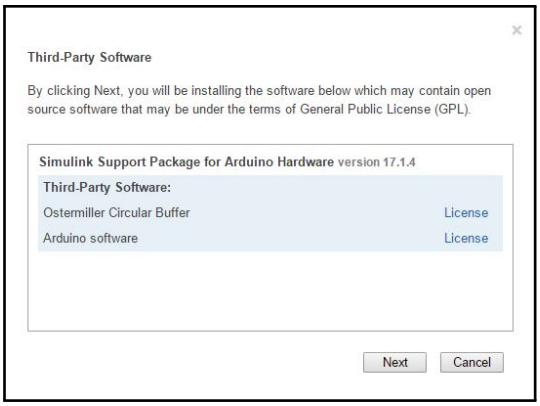



Figure E.32: Clicking Next

34. Click *I Accept* as shown in Figure E.31.

35. Click *Next* as shown in Figure E.32.

36. Next, the installation process will begin, which may take a few minutes. Also, a prompt might pop up about letting it make changes to your computer. Just click *yes*.

37. Finally, a window will pop-up saying that the installation is complete. Uncheck the box that says *Open Examples* and click *Finish* as shown in Figure E.33.

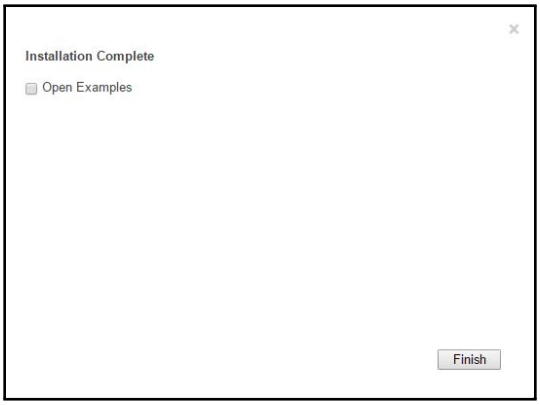



Figure E.33: Finish Installation

**Setting Up Simulink File**

38. With MATLAB R2017a open, type "simulink" in the command window.

39. A Simulink Start Page will open up. Click on *Blank Model* as shown in Figure E.34.

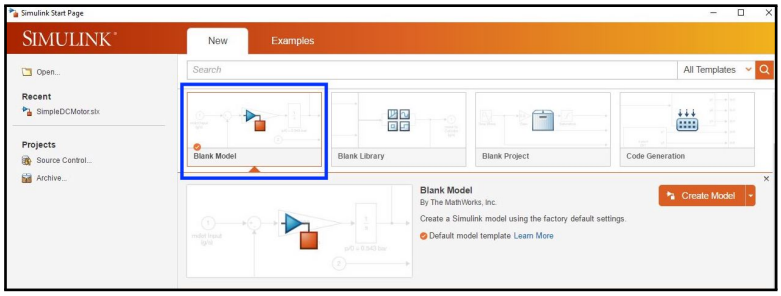



Figure E.34: Simulink Start Page

40. Now find the Library Browser button on the menu bar (below the bar containing file, edit, etc.). Note that the button contains four squares of colors - red, blue and white. Click on the Library Browser button  . The Library Browser should now be visible.

41. At this point, the Arduino Simulink package should be visible under one of the options on the left half side of the Library Browser (the portion with a scroll bar). Click on the drop down menu under *Simulink Support Package for Arduino*. New options should now be visible on the right hand side of the browser.

42. On the right side of the browser, double-click on the block labeled *Common*. (Refer to Figure E.35 to see how it looks in the library browser.) Various Arduino blocks should now be visible. Click, hold, and drag the block labeled *PWM* and drop it onto the blank Simulink model window. PWM or "Pulse Width Modulation" is a series of voltage pulses used to drive many DC motors via digital output. The PWM signal is defined by the frequency of the pulses and the percent of time that the pulse is high (the duty cycle). The Simulink PWM block uses 490 Hz, so only a duty cycle input is required.

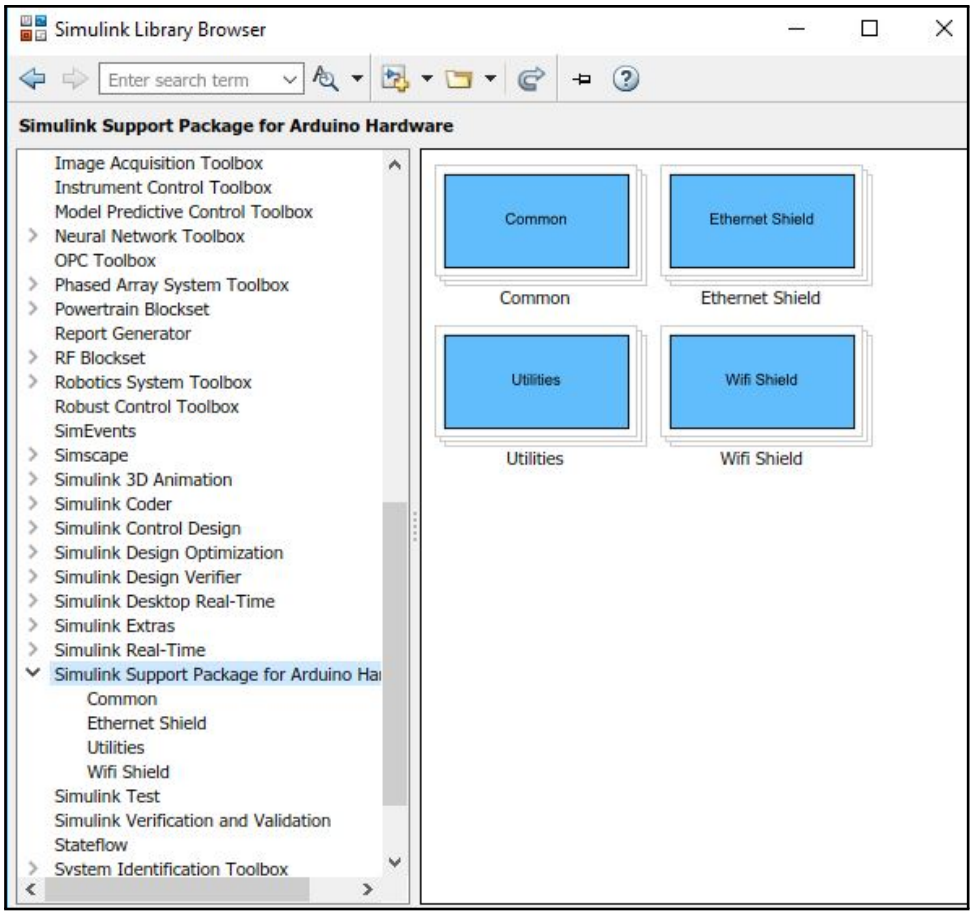



Figure E.35: Simulink Support Package for Arduino

43. Double-click the PWM block and input 5 as the Pin number as shown in Figure E.36. Pin 5 on the motor shield is used to specify the duty cycle of the voltage being applied to the DC motor. Pin 5 accepts a value from 0 to 255, which varies the duty cycle from 0 to 100 percent in the PWM wave.

44. While in the Simulink Library Browser, click, hold, and drag the block labeled *Digital Output* and drop it on to the Simulink model window.

45. Double-click on the Digital Output block and input 4 as the Pin number as shown in Figure E.37. Pin 4 on the motor shield is used to define the DC motor direction. In this case, a 1 on pin 4 specifies the clockwise direction and a 0 specifies the counter clockwise direction.

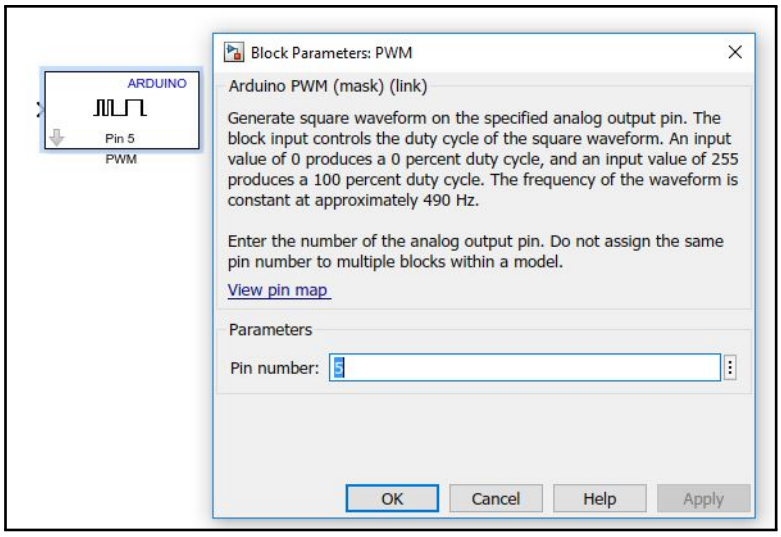



Figure E.36: Setting Up PWM Block

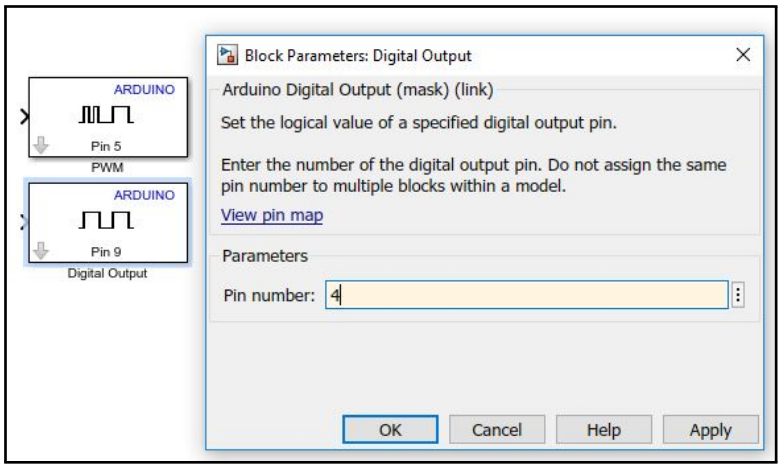



Figure E.37: Setting Up Digital Output Block

46. There should now be two blocks (Digital Output and PWM) in the Simulink model window. In order to find the rest of the blocks used in this experiment, open or bring the Simulink Library Browser into focus once again. Click on the option *Simulink* on the left half portion of the window located at the top of the list. Figure E.38 shows what the library browser should look like.

47. There will be multiple subsections where blocks will be pulled. Some of these subsections include *Commonly Used Blocks*, *Continuous*, *Math Operations*, etc. Drag

4 *Constant* blocks and a *Product* block from the commonly used blocks section onto the Simulink model window as shown in Figure E.39.

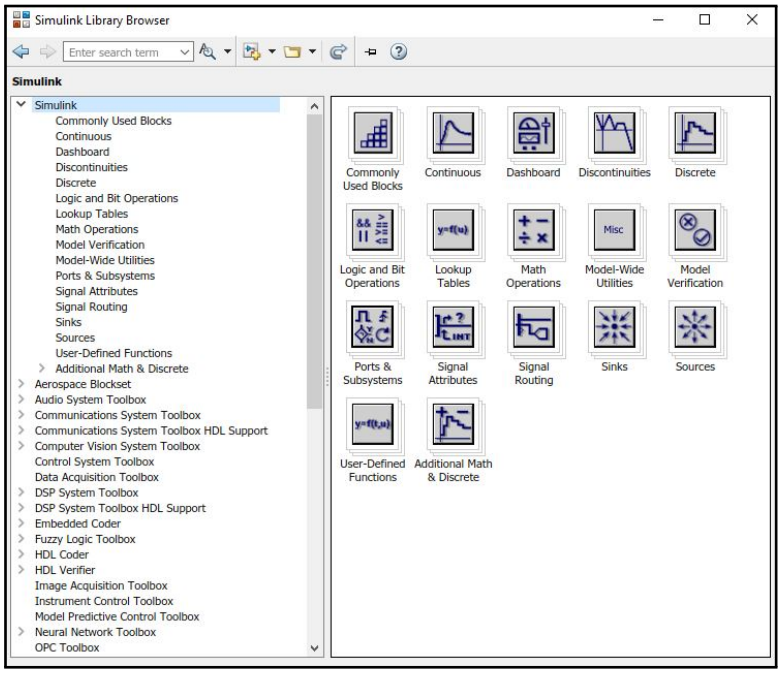



Figure E.38: Main Simulink Blocks in Library Browser

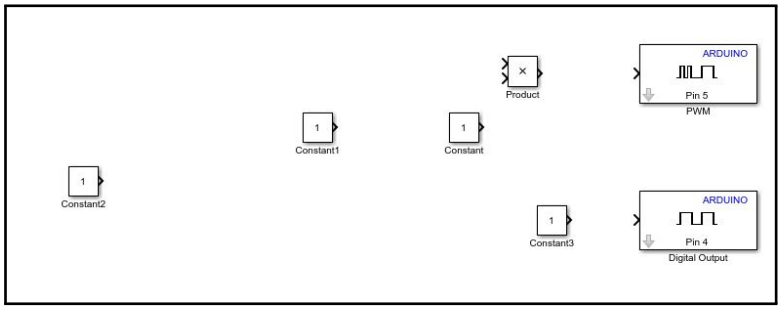



Figure E.39: 4 Constant Blocks and a Product Block

48. In the library browser go to the *Math Operations* subsection and drag in a *Divide* block to the Simulink model.

49. To edit a constant block, double click the block. Figure E.40 shows the values to be placed into the Constant1 block shown in Figure E.39. The Constant1 block will become the *Max Motor Voltage* block. (You can click in the name to edit it.) Set the constant value to 12, and make the sample time 0.03.

50. The rest of the constant values are shown in Figure E.41. All of the sample times for the constant blocks should be set to 0.03.

51. To connect two blocks, take the output of one block and drag the arrow into the input of another block. Figure E.42 shows the final configuration of the model. You have created a Simulink model that will take a voltage value, divide it by 12 (the maximum allowable voltage), and then multiply the result by 255. The result, which is then applied to pin 5 of the Arduino, represents the duty cycle of the PWM. A value of 0 represents a zero percent duty cycle, and a value of 255 represents a one hundred percent duty cycle. You will experiment with this process in a later step.

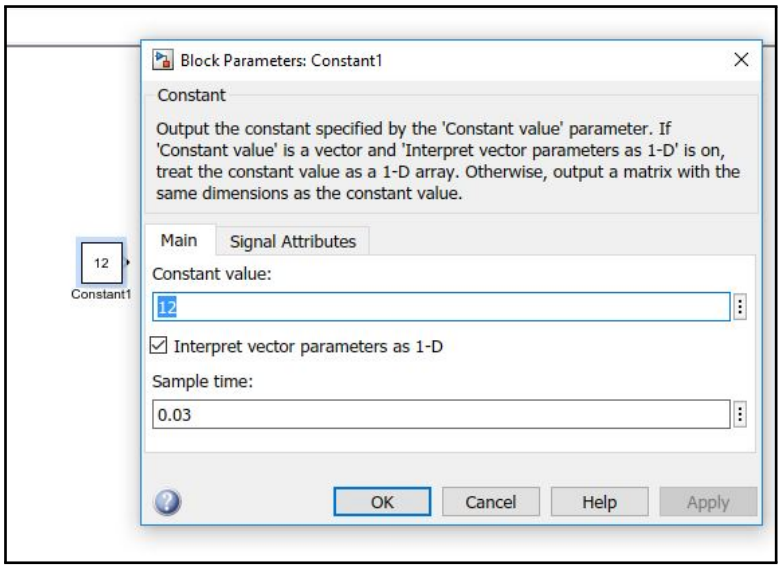



Figure E.40: How to Edit Constant Block

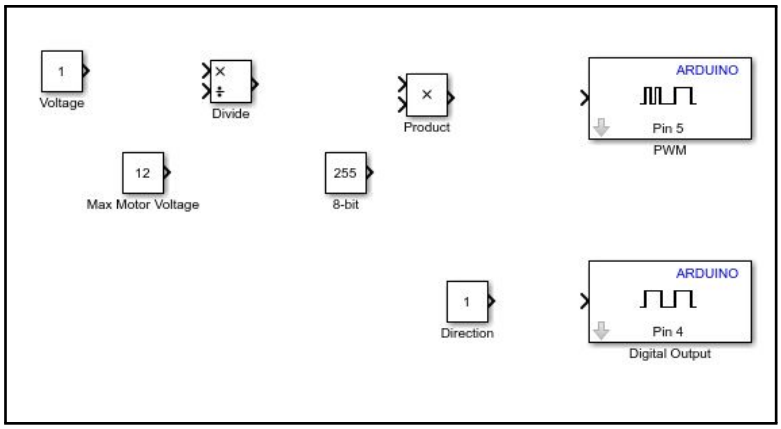



Figure E.41: Updated Constants and Added Divide Block

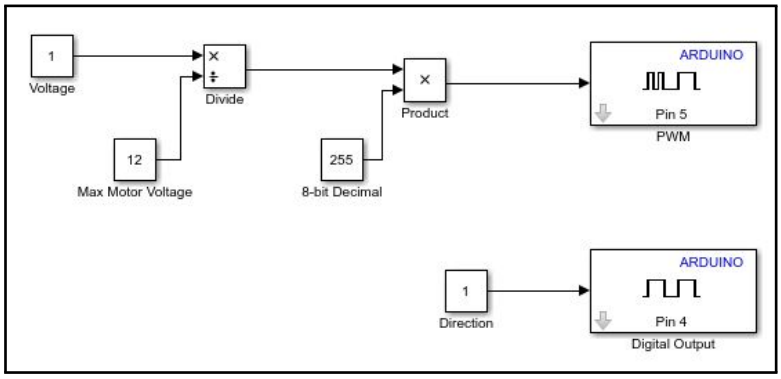



Figure E.42: Final Simulink Model

52. Before continuing to the Configuration Parameter setup, simply plug in the USB-A connector from the Arduino board into the computer running this project.

**Configuration Parameters**

53. Click on *Tools → Run on Target Hardware → Prepare to Run.*

54. On the page labeled "Configuration Parameters: file name/Configuration (Active)" click on the drop down menu next to "Hardware Board:." Choose the option *Arduino Mega 2560* as shown in Figure E.43.

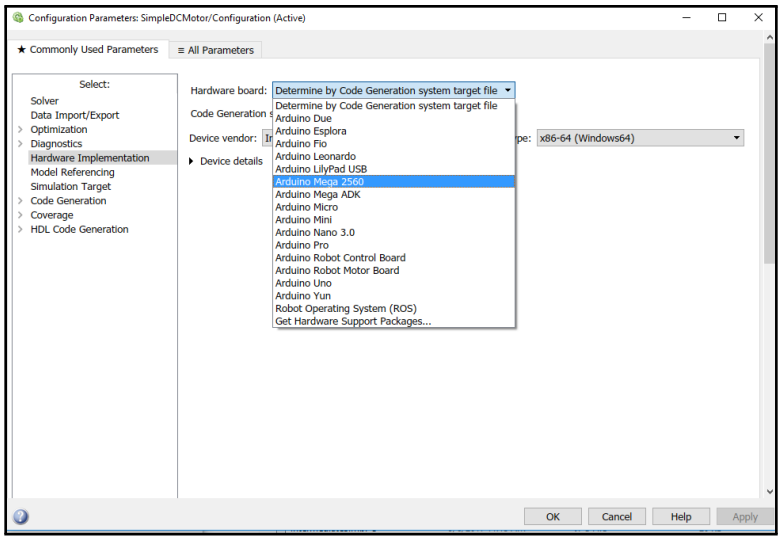



Figure E.43: Set Target Hardware to Arduino Mega 2560

55. Now many more options should appear under the Target Hardware Resources menu. While many of these options are useful in regards to communication, leave them as default. Under *Host-board connection* ensure that the *Set host COM port*: is set to *Automatically*, as shown in Figure E.44. This will allow Simulink to automatically detect the Arduino COM port. (If Simulink is unable to find the Arduino COM port, navigate to the device manager and find out the corresponding COM port for the Arduino Mega 2560. Also set the *Set host COM port*: to Manually and enter the correct COM port).

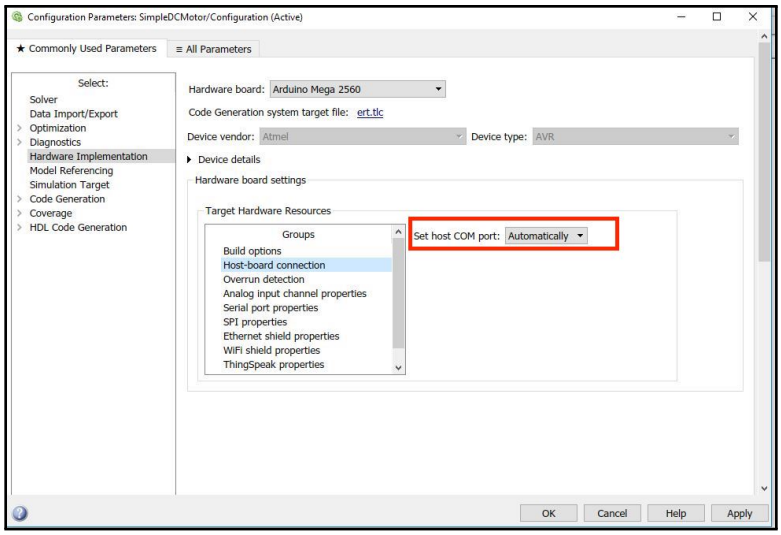



Figure E.44: Set COM Port to Automatic

56. On the left pane, click on the option *Solver*. While on the Solver page, change the *Fixed-step size (Fundamental sample time)*: to 0.03 seconds and the *Stop time*: to inf (meaning infinite). (See Figure E.45.) After making these changes, now click "OK" at the bottom of the page.

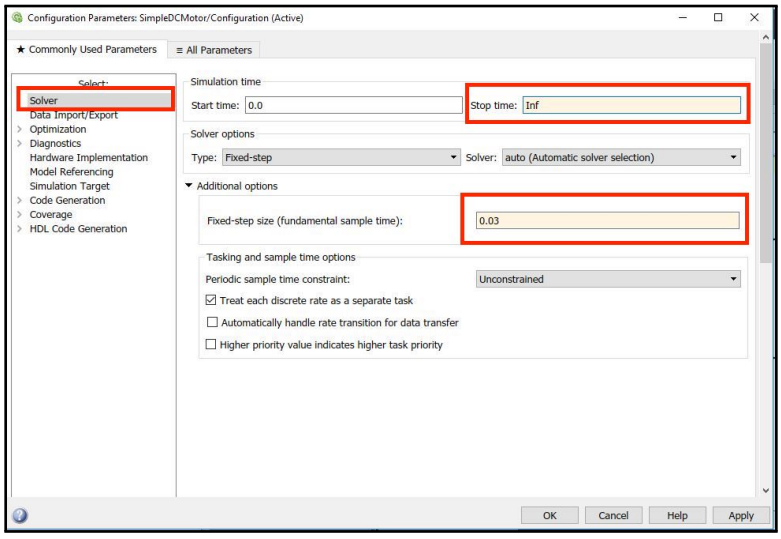



Figure E.45: In Solver Pane, Change time to Inf and step size to 0.03

57. At this point, the model is now ready to run on the Arduino. To deploy the model onto the Arduino, click the *Build Model* button  or Ctrl+B.

58. Sometimes compiling will take a little time. If the model was successfully deployed to hardware there will be a message in the bottom left-hand corner that says *Ready*. If there is an error with deploying to hardware a window that says *Diagnostic Viewer* will pop up with a summary of what the error is.

59. The motor will also need to be powered which is done through the DC power supply. When the model has finished deploying to hardware, the DC motor can be powered.

60. To power the motor, plug the DC power supply into an outlet, and connect the male end of the supply to the female barrel jack that is connected to the motor shield, as shown in Figure E.46. **Note:** it is recommended to unplug the DC power supply from the motor every time something is being uploaded to the board.

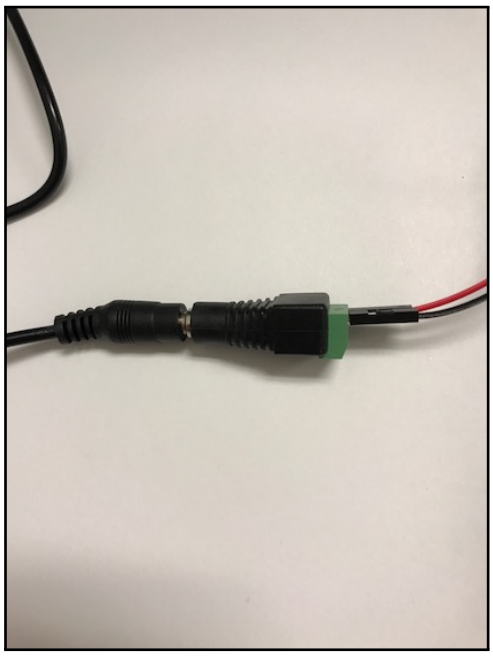


Figure E.46: Connecting the Motor Shield to 12V DC

## E.3 Experimental Procedures

Now that the hardware and software have been set up, the exercises can be performed by loading the Simulink models to the Arduino and running them. There are two ways that Simulink can communicate with the Arduino: *Normal Mode* and *External Mode*. The exercises in this experiment will use normal mode. In normal mode, it is more difficult to transfer data back and forth interactively, but the software on the Arduino can run faster.

### E.3.1 Exercise 1: PWM

Pulse Width Modulation (PWM) is a method for approximating the effect of an analog voltage by switching a constant (digital) voltage on and off quickly. PWM operation is defined by the percentage of time, called the duty cycle, that the signal is high during each period of the oscillation. Another name for the period is the cycle. For example, if a signal has a 50% duty cycle, then it is on for half of the period and off for half of the period. If the duty cycle is 25%, then the signal is on for 25% of the period and off for 75% of the period. For the Arduino, the frequency of this signal is 490 Hz, or period T = $\frac{1}{490}$ seconds. The Arduino output receives an 8 bit digital value, which in decimal represents values 0 to 255. The value 0 represents 0% duty cycle, and 255 represents 100% duty cycle.

The average voltage generated by a PWM wave with amplitude $A$ can be calculated

as:

$$s_{avg}(t) = \frac{1}{T}\int_0^T s(\tau)d\tau$$

*For the 50% duty cycle case*

$$s_{avg}(t) = \frac{1}{T}\Big[\int_0^{0.5T} Adt + \int_{0.5T}^{T} 0dt\Big]$$

$$s_{avg}(t) = \frac{1}{T}\Big[At\Big]\Big|_0^{0.5T}$$

$$s_{avg}(t) = \frac{1}{T}\Big[0.5AT\Big]$$

$$s_{avg}(t) = 0.5A$$

Figure E.47 shows what a 50% and a 25% duty cycle square wave should look like.

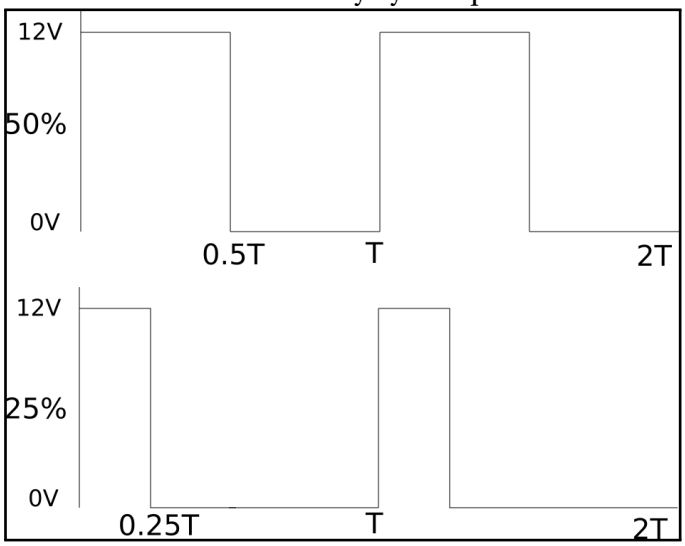



Figure E.47: Example of PWM 25% and 50% Duty Cycle

A motor can be viewed as a lowpass filter (for example, the simple RC circuit shown in Figure E.48), as you will see in later experiments. If the period of a PWM signal applied to a low pass filter is much shorter than the response time of the filter, then the capacitor won't have enough time to fully charge during one period. You will experiment with this concept in the following steps.

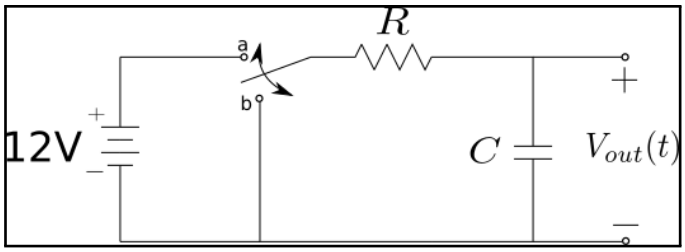



Figure E.48: Simple RC Circuit

61. Derive a transfer function from the input voltage to the output voltage when $R$ = 1Ω and $C = 1F$. The Simulink file *rc.slx* that you downloaded and extracted simulates this transfer function. It will be called by the file *rc_script.m* to simulate the operation with different PWM frequencies in the following steps.

62. From the Matlab command window, open the file *rc_script.m* that you downloaded and extracted earlier. It should open in the Matlab editor.

63. Run the *rc_script.m* file. In the editor, you can click the green run arrow. **Note:** three figures will pop up when the file has completed running.

64. Figure 1 in the MATLAB plots will show a slow PWM frequency (6 sec period), Figure 2 will show a higher frequency (0.5 sec period), while Figure 3 will show a frequency similar to what the Arduino uses for a PWM cycle (0.002 sec period). Examine the three figures to get an understanding of why we need a fast PWM frequency if we want the response to the PWM signal to be similar to the response of a fixed voltage. (In other words, we want the response to a 50% duty cycle PWM signal, with 12 volt maximum, to be the same as a response to a constant 6 volt signal.)

65. Using the transfer function you found earlier, find the response to a 6 volt step function input and plot the response. Compare it to the plot in Figure 3 above. Is the response of the 50% duty cycle 12 volt PWM signal the same as the response to a 6 volt step function?

66. Figure E.42 shows how the 8-bit digital PWM code (from 0 to 255) is computed. Fill in Table E.2 to develop an understanding of the resolution achieved by this process. For each voltage, compute the corresponding duty cycle of the PWM wave (assuming a 12 volt maximum). Then compute the corresponding digital value (from 0 to 255) that will produce the desired duty cycle. In addition, discuss the minimum change in duty cycle that you can acheive with the 8 bit digital code (0 to 255). You will add the description of the motor operation in the next exercise.

Table E.2: Voltage Table

| Voltage | Duty Cycle (%) | Digital Value$_{10}$ | Describe Motor Operation |
|---|---|---|---|
| 0.5 | | | |
| 1 | | | |
| 1.5 | | | |
| 2 | | | |
| 4 | | | |
| 6 | | | |
| 8 | | | |

### E.3.2 Exercise 3: Comparing Duty Cycles with Normal Mode

67. At the top of the Simulink model, where the drop down menu shows either *Normal* or *External*, make sure it is set to *Normal*.

68. Double-click on the Voltage block and change the value to the first value in Table E.2. (Each time the voltage block is changed, you must *Deploy to Hardware* to update the value.)

69. After the project downloads and the motor runs, record how the motor responds in the description column. Does it move? Does it move fast or slow? Does it move at a constant speed? **Note:** every time you adjust the voltage, you need to deploy the model to hardware.

70. Repeat the above steps for each voltage in the table.

71. Change the voltage to 3 volts and redeploy to hardware.

72. Change the value in the Direction block to 1. Click the Deploy to Hardware button. Describe the resulting response. Make note of the direction the motor is moving before and after you changed the direction value.

73. When you apply a very small voltage to the motor, as you found in the previous steps, it will not move. This is because of static friction. Every motor has a different level of static friction, and the static friction level can be different in different directions. Find a voltage threshold for each direction where your motor just begins to spin.

### E.4 Table of Discussions and Questions

Before you turn in your report for this experiment, make sure that you have answered all of the questions that have been posed. It is important that your answers be expansive and that they demonstrate that you were mentally engaged in the experiment. Below is a recap of the important questions and the number of the step where each question was embedded.

| steps | Discussion/Question |
|---|---|
| 61 | Transfer function of RC network |
| 64 | Discussion on different PWM frequencies |

| | |
|---|---|
| 65 | Comparison of theoretical step response and PWM response |
| 66 | Voltage resolution due to digital representation |
| 68 | Fill out the voltage table |
| 69 | Descriptions of motor operation |
| 72 | How does direction line affect motor operation? |
| 73 | Friction threshold for each direction |

## E.5 Conclusion

This experiment provided a basic introduction to the use of the Arduino to control a DC motor. An introduction was provided to the "normal mode" operation, in which software is downloaded to the Arduino and runs outside the Simulink environment. This experiment forms the foundation for many future experiments, as the concepts presented here are used in almost all other labs.

# APPENDIX F

## Sampling and Data Acquisition Handout

### F.1 Objective

This experiment will review the concept of sampling, and will show how to acquire data in real-time, at multiple sampling rates, using Simulink. Normal Mode is the method of collecting data in real-time from Simulink in this lab. The experiment is designed to show key concepts of sampling to get a better understanding of how sampling can be applied to physical systems. You will use an Arduino to collect data from a DC motor. You will get hands-on experience with Normal Mode.

### F.2 Setup

#### F.2.1 Required Materials

**Hardware**

- All materials from Simple DC Motor experiment

**Software**

- Matlab/Simulink 2017a
    - ⋆ *The steps and images related to Matlab/Simulink for this experiment were created using Matlab/Simulink 2017a. Therefore some steps and images may be a little different if you are not using this version. If you are in fact using a different version, make sure you know the steps for running models onto the*

*Arduino for your version of Simulink.*

- Matlab/Simulink files
- Take Home Labs Arduino Library

**Prerequisite Experiments**

The experiments listed below should be completed prior to this one. Steps and hardware from these experiments may be needed or referenced throughout this experiment. It is assumed that you already have the Arduino communicating with the PC, know how to set up and run models in Normal Mode, can troubleshoot any of the issues discussed in the lab, and can 3-D print the required parts.

- Simple DC Motor

### F.2.2 Software Setup

The following steps cover the process of downloading and creating the necessary files needed for this experiment. The first section covers downloading the required Matlab/Simulink files from the website, and saving them in the Matlab directory that will be used. Downloading and adding additional Simulink blocks to the Simulink Library is covered in the second section.

**Matlab/Simulink Files**

1. Open your internet browser and navigate to `thl.okstate.edu`
2. On the left side of the Homepage, select “Experiments"
3. In the middle section of the Experiments page select “All"
4. In the middle section of the All Experiments page find and select *Sampling and Data Acquisition*

5. On the Sampling and Data Acquisition page select “Software/Code" in the right-most section. Download the zipfile named *Software_Files*.

6. Right-click the file and choose “Extract All...", or use any other method of extracting the files on your PC.

7. You will now need to set the path for these files, and you should extract the folder somewhere such as the Downloads or Documents folder. Make sure the box next to “Show extracted files when complete" is checked, as shown in Figure F.1. Now select “Extract"

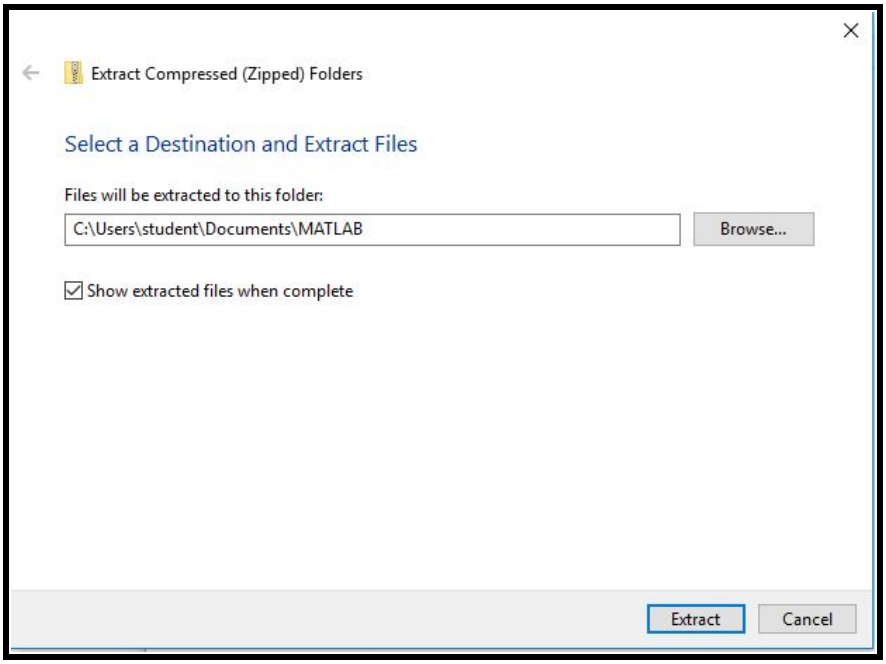



Figure F.1: Zipped Folder Extract

8. A new window should open. Double-click the *Software_SampExp* folder and ensure the zipfiles in Figure F.2 are shown

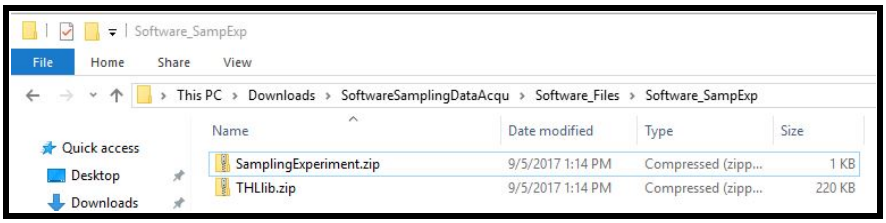



Figure F.2: Software Files

9. Right-click the *SamplingExperiment* folder, and select “Extract All..."

10. Set the path for the folder to be extracted somewhere other than the current location, and select “Extract". **You will need to know this folder location later in the Experimental Procedures section, so you may want to write it down**. You can close the folder if it opens up after extracting it.

11. Leave the *Software_SampExp* folder that contains the zipfiles open, and proceed to the next section.

### Additional Arduino Support Package

Many of the Take Home Labs experiments listed on the website require some type of hardware, other than the Arduino. The hardware may include DC motors, encoders, motor drivers, ultrasonic sensors, etc. In order to make it easier to interface with these hardware components, there are custom Simulink S-Function blocks that have been created. Essentially, the S-Function is just a Simulink block representation of some code. More information about S-Functions can be found on the Mathworks website. The Arduino coding language is just a set of C/C++ functions, and the S-Function blocks you will download contain the code that is needed to interface with these hardware elements. This eliminates the need to to write C code, and it also helps keep the Simulink model organized.

The S-Function blocks that will be needed throughout the experiments will all be

contained in one location in the Simulink Library Browser. It should be mentioned that the additional library that will be added is a modified version of the Rensselaer Arduino Support Package (RASPlib). Their library is available on their website `http://homepages.rpi.edu/~hurstj2/` along with their own curriculum dedicated labs for their own specific hardware that you can purchase. The concept of their labs being dedicated to a curriculum and run with the Arduino and Simulink is similar to the concept of our take home labs. This made their labs a good guide for working with the Arduino and Simulink. Therefore, after learning how to use their library, some of the custom blocks and files they created were modified to work with our experiments and hardware. More information about their labs and what they do can be found on `http://minseg.webs.com/`. Follow the steps below to add the additional support to Simulink needed for this experiment.

12. Open Matlab, and in the Command Window type *pwd* to get Matlab's home directory on your PC. Copy the location or write it down.

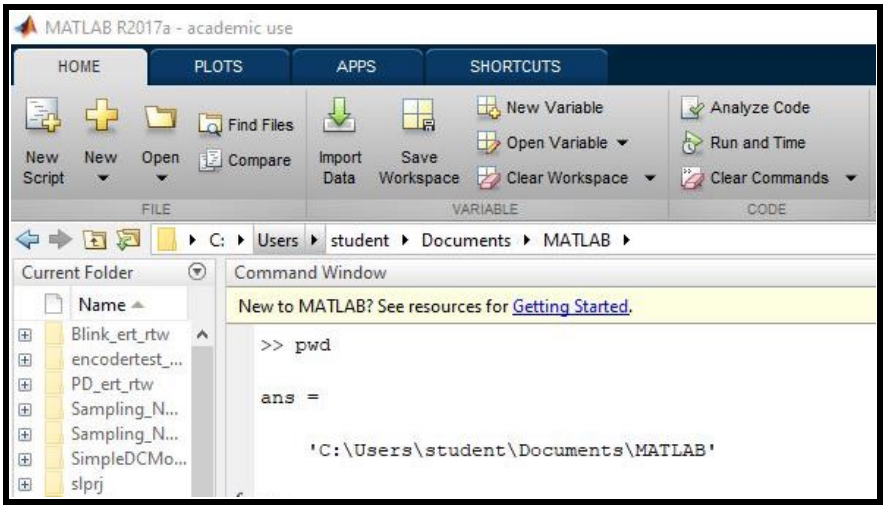



Figure F.3: Matlab Home Directory

13. In the *Software_SampExp* folder, shown in Figure F.2, right-click the *THLlib* folder and choose "Extract All..."

14. Extract the files to the location of your Matlab home directory. You can paste the directory if you copied it, browse and find it, or manually type it in.

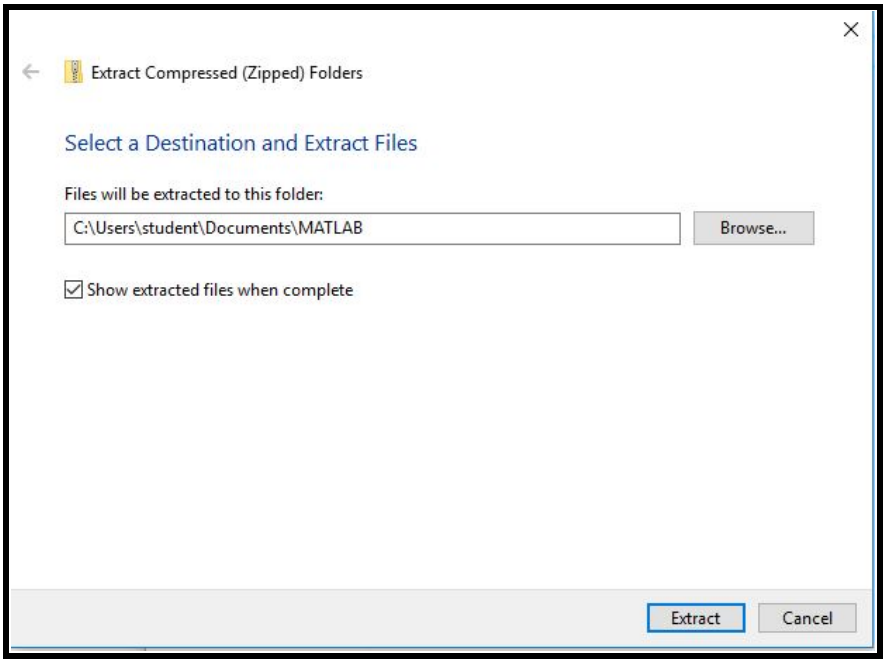



Figure F.4: Support Folder

15. The *THLlib* folder should now be in the Matlab home directory. There should also be a Matlab code file named “startup" in the same location, as shown in Figure F.5. If the file does not exist, navigate back to Matlab.

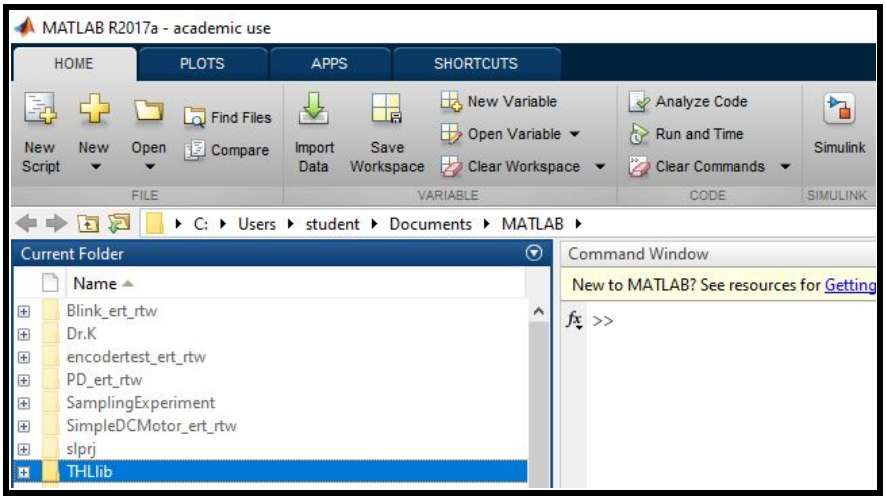
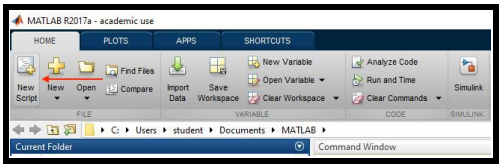
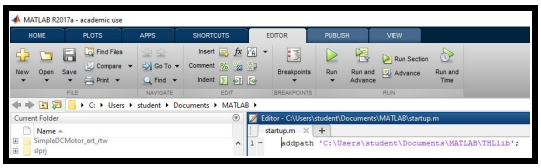



Figure F.5: Directory Folder

16. In Matlab, select “New Script"  , and in the script type your path to the *THLlib* folder, as shown in Figure F.6b for my path. Make sure to put the single quotes around the folder path.




(a) New Script




(b) Startup File

Figure F.6: Creating Startup File

17. Save the file as *startup*. This script will execute every time Matlab starts, and will add the path to the *THLlib* folder each time automatically. Close and reopen Matlab.

18. After you reopen Matlab, type *simulink* in the Matlab Command Window. Make sure the Library “Take Home Labs Arduino Support Package" is now shown in the Simulink Library Browser. If it is not shown, follow the instructions described in the sub-steps below.

    (a) The Simulink library “Take Home Labs Arduino Support Package" was created with version R2013a. Newer versions will need to fix the Simulink environment so that the the older library will show. In the Simulink Library Browser window, press F5 or View → Refresh Tree View.

    (b) After the library refreshes, the message “Some libraries are missing repository information. Fix" will appear at the top of the window as shown in Figure F.7. Press “Fix".

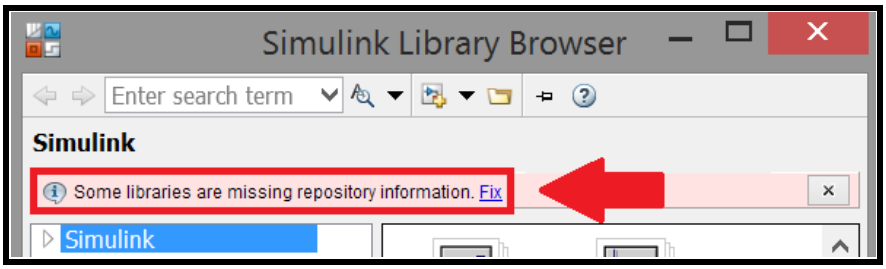



Figure F.7: Simulink THL Library Error

(c) The window shown in Figure F.8 should appear. Select “Resave libraries in SLX file format" and press OK.

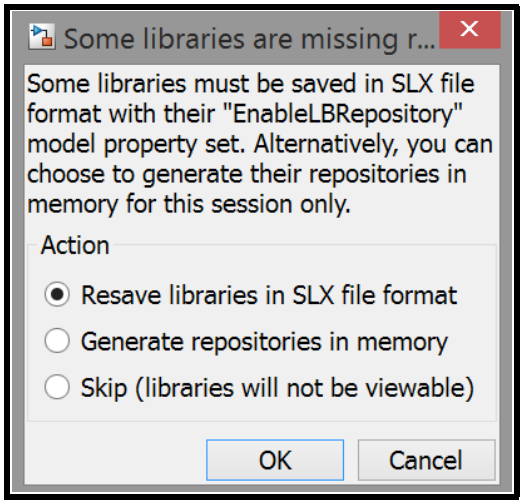



Figure F.8: Simulink THL Library Error (2)

(d) Once the library is fixed, the “Take Home Labs Arduino Support Package" should now show, as in Figure F.9. If it does not show, try closing and reopening Matlab. Then recheck to see if the library shows by navigating back to the Simulink Library Browser.

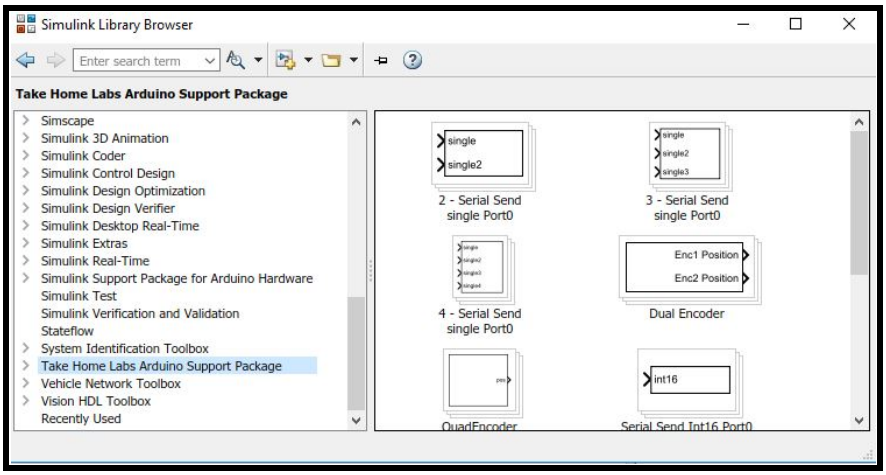



Figure F.9: Simulink THL Library

### F.2.3 Hardware Setup

There is no additional hardware setup needed to be done for this experiment. Please refer to the Simple DC Motor experiment for all hardware related steps.

## F.3 Experimental Procedures

Digital systems, such as a PC and the Arduino, have limitations on how fast they can perform tasks. Sampling is the process of creating a discrete-time signal from a continuous-time signal. The value of a continuous signal is sampled, or measured, at certain time intervals. Depending on the continuous signal being sampled, there are requirements on how fast you need to sample in order to preserve the information in the signal. The first and second exercises of this section will discuss sampling theory, using Simulink to help illustrate the idea. This will be done by sampling a sine wave at different rates and observing the output response. The third exercise of this section will explore sampling with the Arduino using Normal Mode. A sine wave voltage will be input into the motor. Since the motor is a linear system, the output position and velocity should be sine waves of the same frequency. The amplitude and phase of the output sine waves will change with frequency. Various sampling rates will be tested to provide an idea of

how fast or how efficiently the Normal Mode samples data.

Once these experimental procedures are complete, you will have a better understanding of what sampling is, and how to select a good sampling rate for an input of a certain frequency. Also, you will know how to acquire data in Normal mode

### F.3.1 Exercise 1: Sampling Theory Exercise

To help illustrate what sampling is, look at Figure F.10 below. The signal at the top of the figure is one cycle of the sine wave, $A\sin\left(2\pi f t+\phi\right)$. $A$ is the amplitude, $f$ is the frequency in Hz, and $\phi$ is the phase. **Note**: $\omega = 2\pi f$ is another way to represent frequency. The unit for $\omega$ is radians per second [rads/s].

As shown in Figure F.10, $T$ equals the period of the signal. The period of the sine wave is the time it takes to complete one full cycle. The signal at the bottom of the figure shows the same sine wave, but it is sampled every $T_s$ seconds, which is the sampling period. This means that every $T_s$ seconds, the value of the top signal in the figure is being recorded. The sampled sine wave shown in the figure has a sampling frequency of $16/T$ Hz, which corresponds to a sampling period of $0.0625T$ seconds.

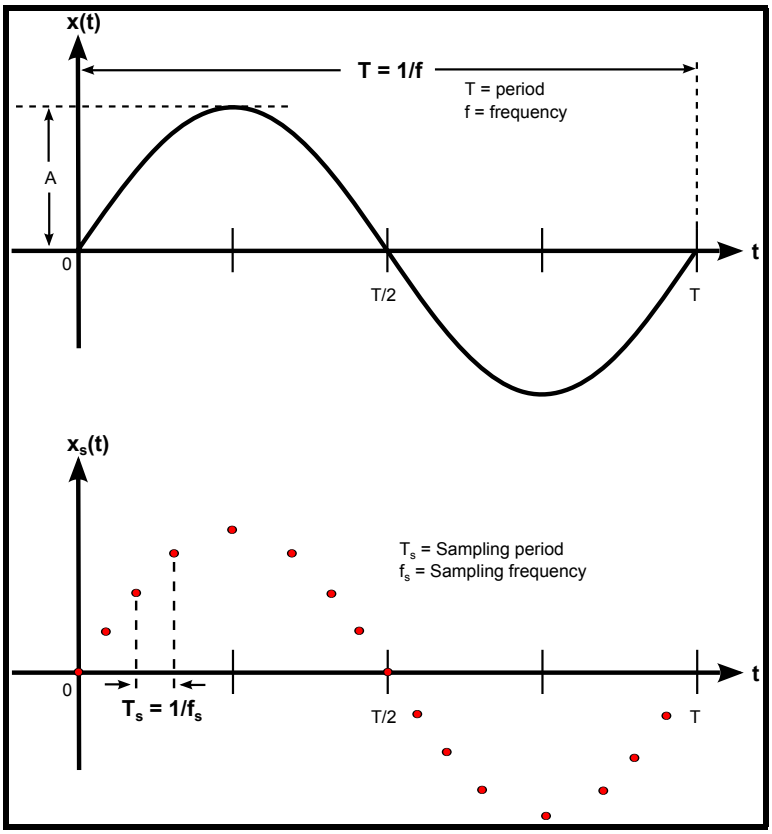



Figure F.10: Continuous and Discrete Sine Wave

The Nyquist Sampling Theorem states that a band-limited signal can be recovered if the sampling frequency $f_s$ is greater than twice the highest frequency of the signal being sampled. This is shown in Equation F.1, where the term $2f_{max}$ is known as the Nyquist Rate.

$$f_s > 2f_{max} \tag{F.1}$$

If the signal is not sampled at greater than the Nyquist Rate, then a phenomenon known as aliasing occurs. What this means is that higher frequency components will be aliased or folded in and appear as lower frequency components in the output signal. This would prevent the signal from being reconstructed properly. The following steps of this subsection will help illustrate this idea.

19. Hand sketch a sine wave with a frequency of 1 Hz ($2\pi$ rad/s), an amplitude of 1, and a phase of 0 for two cycles.

Table F.1: Sine Waves to Sketch

| Sine Wave Frequency f | Sine Wave Period T | Sampling Frequency $f_s$ | Sampling Interval $T_s$ |
|---|---|---|---|
| 1 Hz | 1 second | 16 Hz | 0.0625 seconds |
| 1 Hz | 1 second | 8 Hz | 0.125 seconds |
| 1 Hz | 1 second | 4 Hz | 0.25 seconds |
| 1 Hz | 1 second | 2 Hz | 0.5 seconds |

20. Now use Table F.1 to hand sketch a sampled version of the sine wave at the corresponding sampling frequencies/intervals.

**Note:** before moving onto the next section make sure you have completed the previous two steps and incorporated these sketches in your project notebook. It is crucial to have this done as it will reinforce the idea of sampling theory and will make it easier to understand future experiments.

### F.3.2 Exercise 2: Sampling Theory with Simulink

21. Open Matlab, and select the “Browse for folder" icon

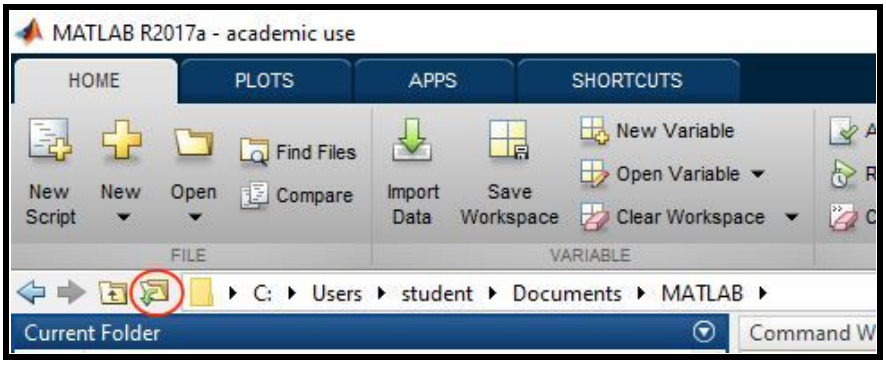



Figure F.11: Browse for Folder

22. Navigate to the location where you extracted the *SamplingExperiment* folder in the software setup section, and press the “Select Folder" button.

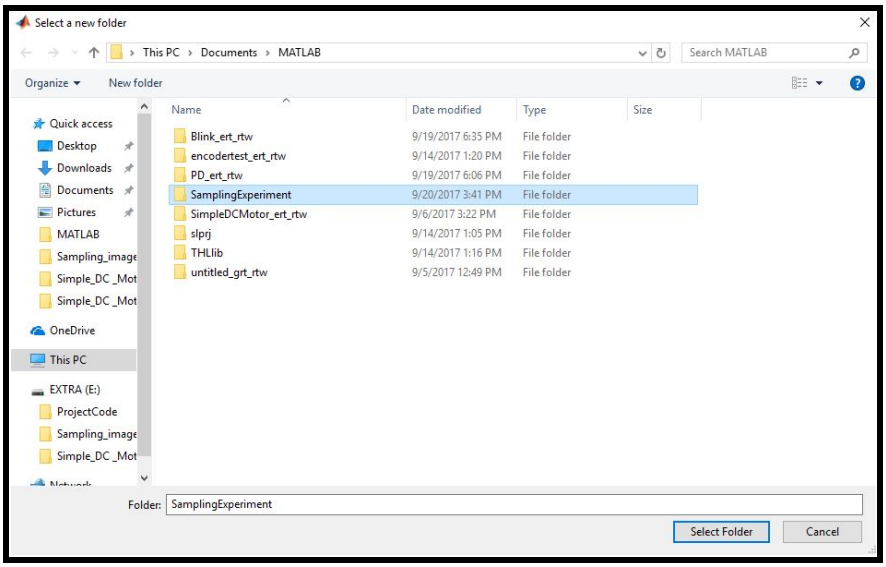



Figure F.12: Experiment Folder Select

23. Make sure Matlab's "Current Folder" section shows the folder you selected, and the "DCmotorDATA.m" file is in the directory.

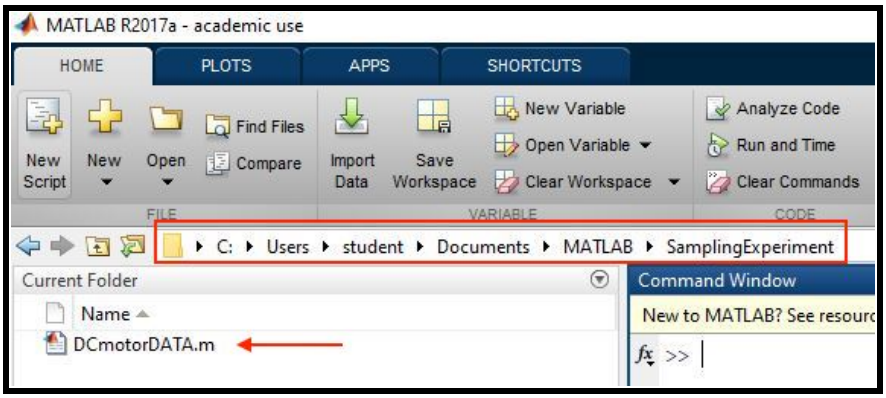



Figure F.13: Current Matlab Directory

24. In the Matlab window, select New → Simulink Model

25. Choose File → Save As. Set the filename to *Exp2Sampling*, and select Save. "Exp2Sampling.slx" file should now be in Matlab's "Current Folder" section.

26. In the Simulink Library Browser add a sine wave to the model. (Simulink → Sources → Sine Wave.) Add a Scope to the model. (Simulink → Sinks → Scope.) Then con-

nect to the two blocks as shown in Figure F.14

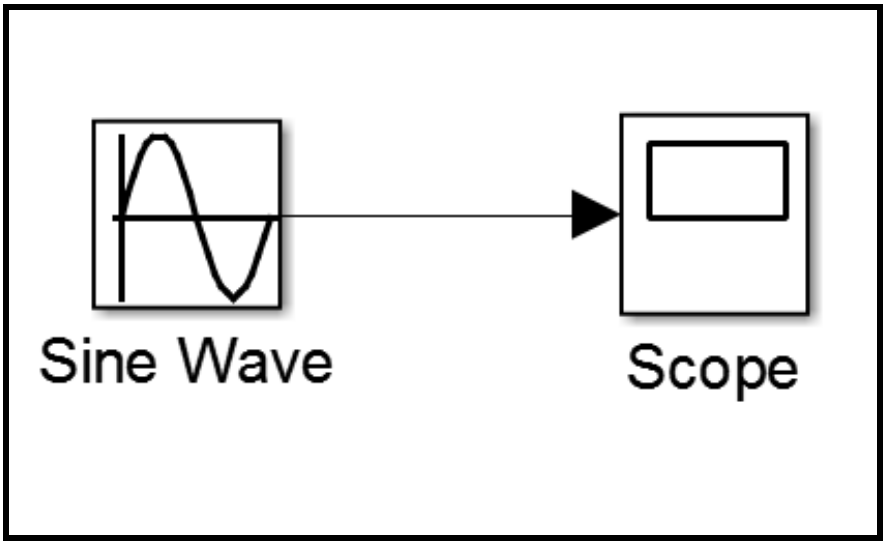



Figure F.14: Sampling Experiment Simulink Model

27. Double-click the Sine Wave block, and change the “Frequency" to *2*pi*. (This corresponds to a 1 Hz sine wave, since $2\pi$ radians/s is 1 Hz.) Select OK.

28. Double-click the Scope block, and select “Parameters"

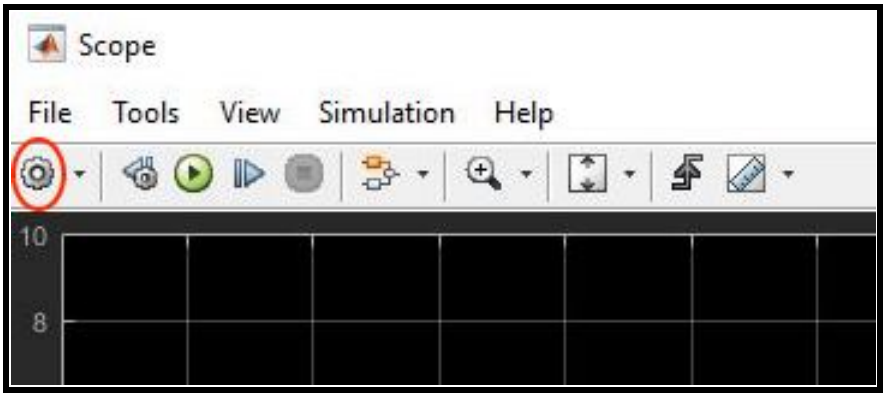



Figure F.15: Scope Parameters

29. Select the Display tab, and set the Y-limits to -1.2 and 1.2 respectively as shown in Figure F.16, then click Apply and OK.

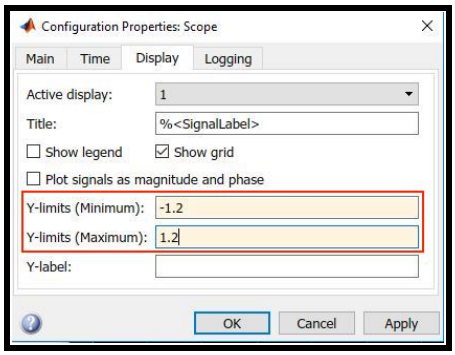



Figure F.16: Scope Y-limits

30. Next in the scope toolbar click View → Style.

31. Then change the "Line" to "no line" and the "Marker" to the circle as shown in figure F.17 and click Apply then OK.

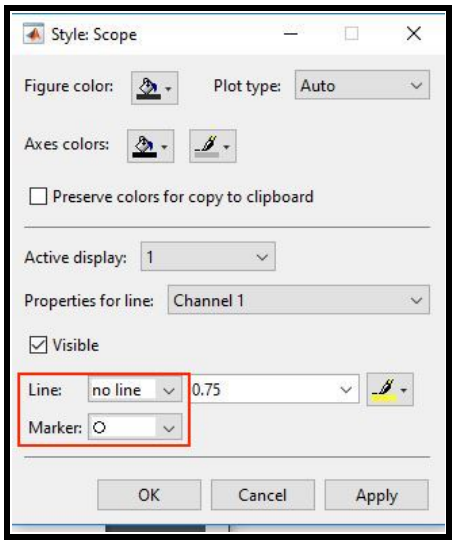



Figure F.17: Scope Line Style

32. Then click on the Model Configuration Parameters in the simulink file  .

33. Set the "Stop time" to 2, the "Solver Type" to "Fixed-step", and the "Fixed-step size" to 0.0625. This step was covered in the Simple DC Motor lab. **Note**: The fixed step size corresponds to the sampling interval.

34. In the Sine Wave block, the Amplitude is set to 1, Frequency $\omega$ is $2\pi$, and the phase is 0. How fast would you need to sample it to avoid aliasing?

35. In the Simulink model, select Run  .

36. Double-click the Scope to pull up the plot.

37. Complete the plots for the cases in Table F.2 by repeating steps 32 through 36. The sampling rate is set by changing the fixed-step size, as in 33.

38. Compare the plots generated via Simulink with your hand sketches. Are the plots what you expected? Why or why not?

Table F.2: Sine Waves to Plot in Simulink

| Sine Wave Frequency f | Sine Wave Period T | Sampling Frequency $f_s$ | Sampling Interval $T_s$ |
|---|---|---|---|
| 1 Hz | 1 second | 16 Hz | 0.0625 seconds |
| 1 Hz | 1 second | 8 Hz | 0.125 seconds |
| 1 Hz | 1 second | 4 Hz | 0.25 seconds |
| 1 Hz | 1 second | 2 Hz | 0.5 seconds |

39. Ignoring the point at 2 seconds (because it is the start of the third cycle of the sine wave.), how many points are shown on the plots? How has decreasing the sampling rate changed the outputs? Do you think that the original continuous sine wave could be reconstructed from each of the sampled signals?

40. Now, go back and change Fixed-step size to 0.9 seconds and the end time to 10 seconds and run the Simulink model again. What does the frequency of this signal seem to be? Is this what it should be? If not, explain. Can the original continuous sine wave be reconstructed from the sampled signal?

41. Save the model and close it.

### F.3.3 Exercise 3: Sampling and Data Acquisition in Normal Mode

Now that the concept of sampling has been introduced, the next task is to sample with the Arduino in Normal Mode. This section will use all of the required hardware, and it is assumed that you know the entire process for running Simulink models on the Arduino using Normal Mode. This process was described in the *Simple DC Motor* experiment.

42. From the Matlab window, open a new Simulink Model: New → Simulink Model

43. Save the file as *Sampling_NM*. The final Simulink Model will look like Figure F.18

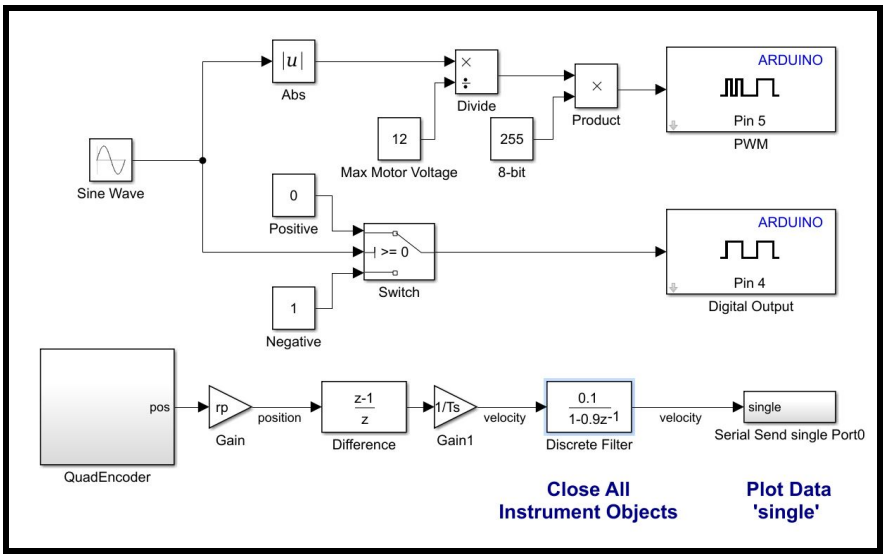



Figure F.18: Simulink Model for Normal Mode

44. Open the Simulink Library Browser  , and add the following blocks to your empty Simulink Model:

    - Sine Wave: Sources → Sine Wave
    - Constant: Sources → Constant
    - Switch: Signal Routing → Switch
    - Abs: Math Operation → Abs

- Divide: Math Operations → Divide
- Product: Math Operations → Product
- Arduino Digital Output: Simulink Support Package for Arduino Hardware → Digital Output
- Arduino PWM: Simulink Support Package for Arduino Hardware → PWM

45. Double-click the Sine Wave block, set the Sine Type to “Time based", Amplitude to 4, Frequency to *2*pi*, and Sample time to *Ts*. Press OK to exit.

46. Double-click the Constant block and change the Constant value to 12, or to whatever is the voltage of your power supply. Also set the Sample time to *Ts*. **Note:** for every constant block for every experiment the sample time will also be the sampling rate of the system. Press OK.

47. Double-click on another Constant block and change the Constant value to 255. Again, change the Sample time to *Ts*.

48. Set the PWM block to Pin 5.

49. There should be two more Constant Blocks, change one of their values to 0 and the other 1. Again, make sure their Sample times are set to *Ts*.

50. Double-click the Digital Output block and change the Pin number to 4. Press OK.

51. Connect the current blocks as in Figure F.19

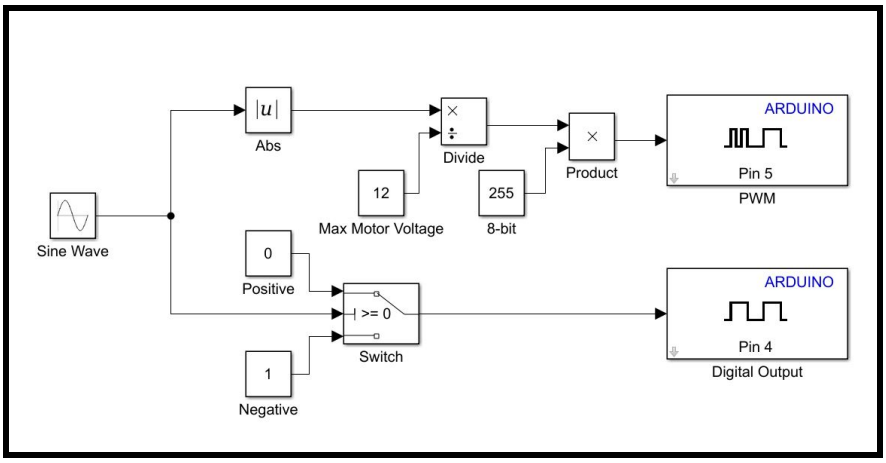



Figure F.19: Simulink Model Setup - Normal Mode (1)

52. Add the following blocks to the Simulink Model:

    - Quad Encoder Block: Take Home Labs Arduino Support Package → QuadEncoder
    - Gain: Commonly Used Blocks → Gain
    - Difference: Discrete → Difference
    - Discrete Filter: Discrete → Discrete Filter
    - Serial Send single: Take Home Labs Arduino Support Package → Serial Send single Port0

53. Change the first Gain block to *rp*.

54. Change the second gain block to $1/Ts$.

55. Double click on the Discrete Filter block. Change the Numerator to *[0.1]* and change the Denominator to *[1 -0.9]* and the Sample time to Ts.

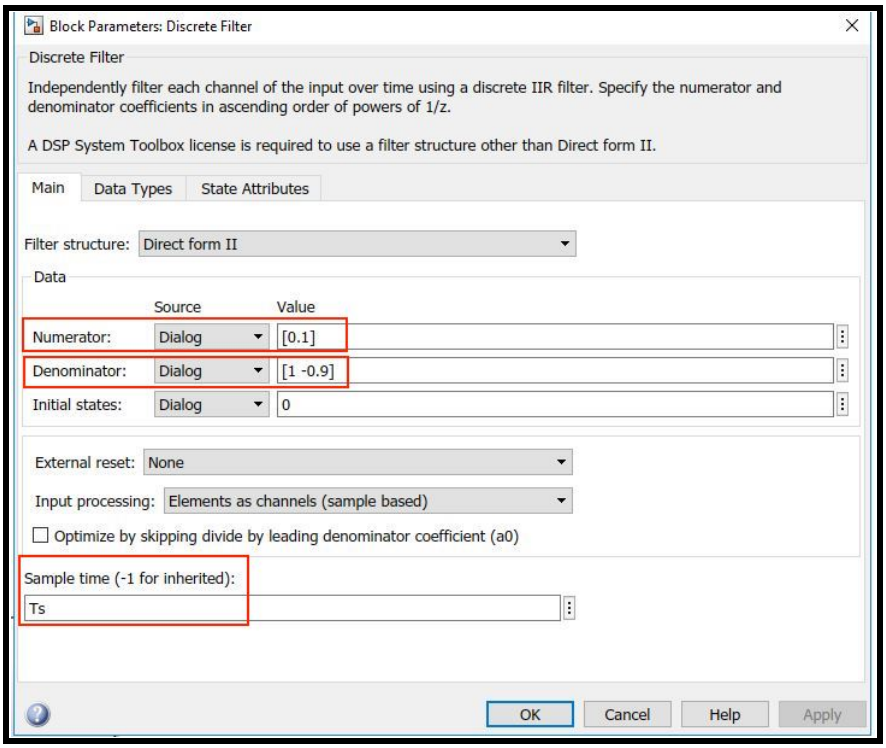



Figure F.20: Discrete Filter Parameters

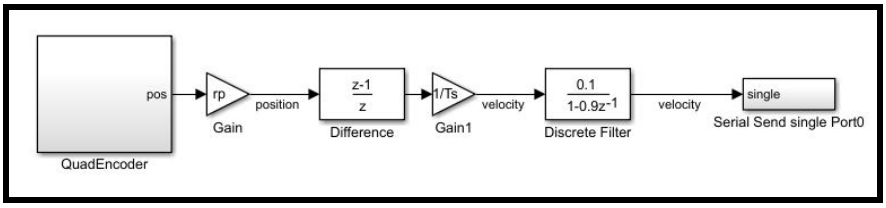



Figure F.21: Simulink Model Setup - Normal Mode (2)

56. Finally, add the text blocks to the Simulink Model. In the Simulink Library Browser, right-click “Take Home Labs Arduino Support Package", then select “Open Take Home Labs Arduino Support Package library". A window similar to that shown in Figure F.22 should open. Click and drag “Plot Data ’single’" and “Close All Instrument Objects" to the Simulink File. The final Simulink model should now resemble the model shown in Figure F.18. The upper part of the figure shows the PWM section of the model, as was used in the Simple DC Motor experiment. The lower part of the figure implements the measurement of motor velocity. When the

motor spins, the encoder sends out a series of pulses (64 per revolution) that are counted and scaled to produce a position measurement. Successive positions are differenced and divided by time to produce a velocity measurement. This measurement is somewhat noisy, so it is put through a filter to smooth the result.

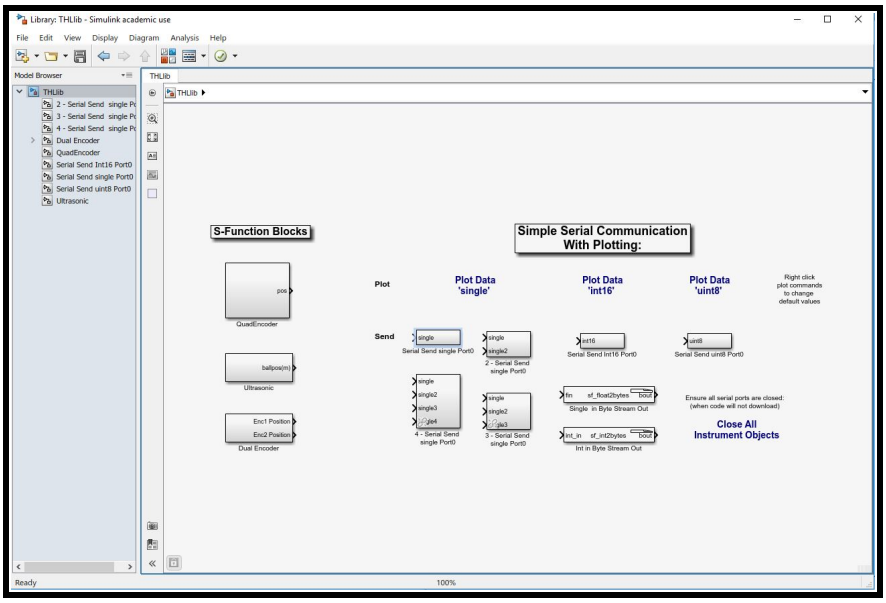



Figure F.22: Simulink Model Setup - Normal Mode (3)

57. Set up the model to run on the Arduino in Normal Mode. Set the Sampling Time (Fixed Step Size) to *Ts*. (Refer to the Simple DC Motor experiment for the procedures to set the step size and operation mode.)

58. In the Matlab Command Window, type *Ts = 0.0625;*. “Ts" should now be a variable in the Matlab Workspace.

59. Also in the Matlab Command Window, type *rp = (2*pi)/64;*. "rp" should now be a variable in the Matlab Workspace.

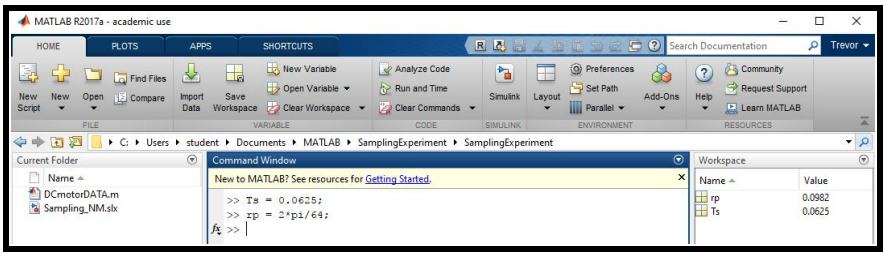



Figure F.23: Sampling Time - Normal Mode

60. Next, connect the Arduino to the PC, and load the model onto the Arduino. (Refer to the Simple DC Motor experiment for procedures.) Remember that in Normal Mode, once you download the model to the Arduino, the code will begin running automatically. Therefore, do not connect the power supply just yet. *Note:* If you receive an error stating that Simulink cannot connect to the board, try to disconnect and reconnect the USB cable.

61. Once the model is successfully downloaded to the Arduino, double click the “Plot Data ’single’" text in the Simulink model.

62. A window, “Plot Serial Data", should open. Change the COM port to the port your Arduino is using. (See the Simple DC Motor experiment to see how to find the port.) Set the number of samples to plot to 64, as shown in Figure F.24. Then hit OK.

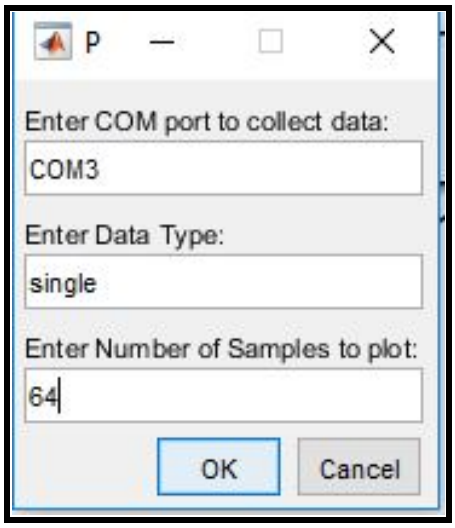



Figure F.24: Plot Serial Data

63. A figure should open up that will start plotting the serial data, as shown in Figure F.25. It may take a few seconds to open.

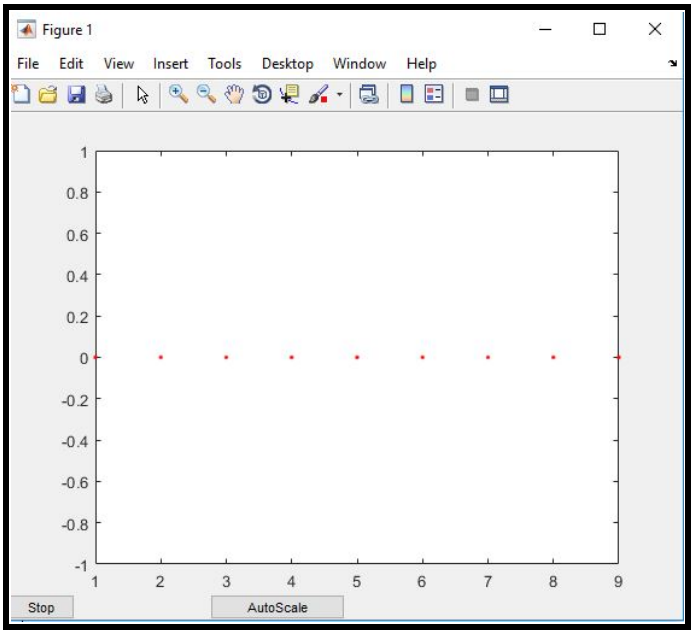



Figure F.25: Serial Plot Window

64. Slightly move the load to the left and right, and then press the “Autoscale" button. The data in the plot should look similar to Figure F.26. The plot shows the angular velocity of the motor versus the number of samples. Recall that a sample takes

place every $Ts$ seconds. Currently, $Ts = 0.0625$.

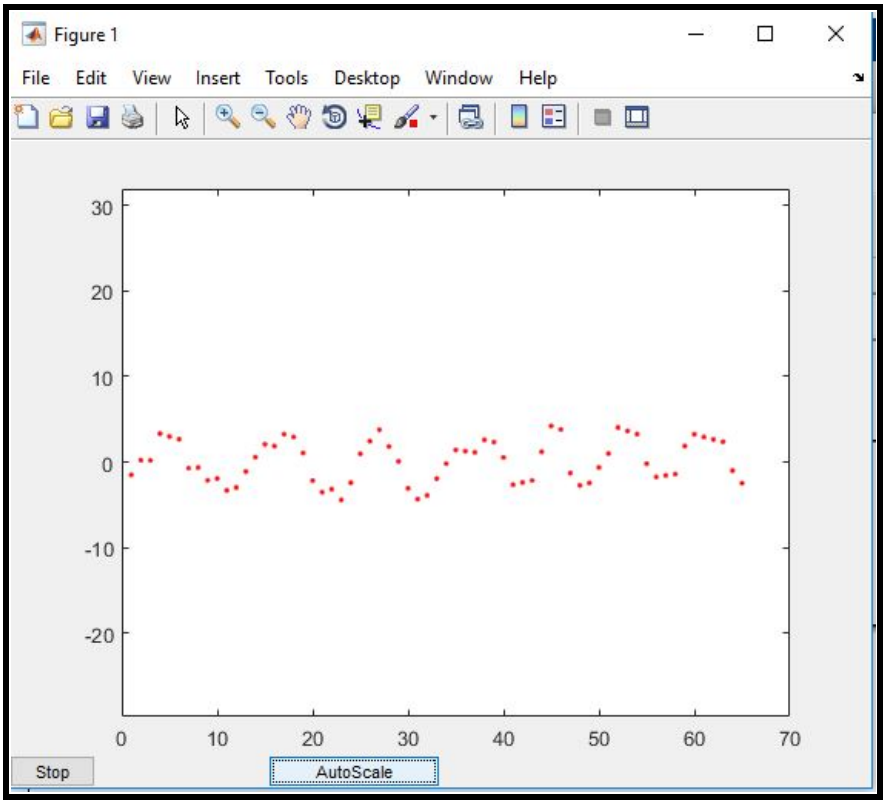



Figure F.26: Serial Data Sync Test

65. Before plugging in the power, make sure the load is attached to the motor shaft securely, and that your hands and any other objects are away from the spinning load, to avoid potential injuries.

66. Plug in the power supply. ***Caution:*** The motor will start to spin the attached load when you connect the power supply.

67. The plot should now show a sine wave similar to Figure F.27. You may need to press Autoscale while the load is spinning to fit the plot axes to the data. The output plot you want should have about 4 cycles of the sine wave, since there are 64 samples being plotted, there are 16 samples per second and the sine wave is 1Hz. When the plot shows 4 cycles of the sine wave, press the “Stop" button located at the bottom left of the plot window.

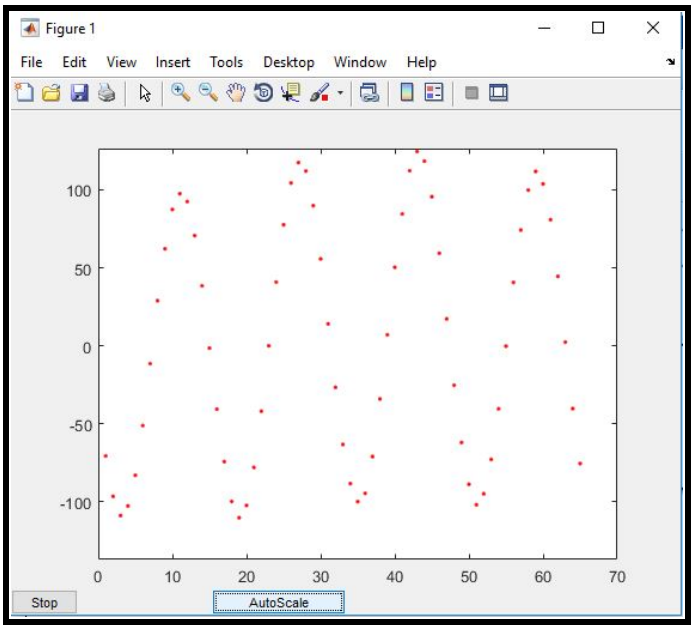
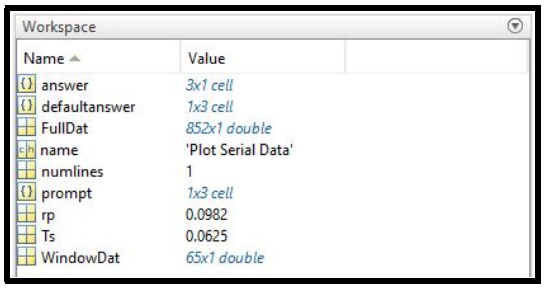


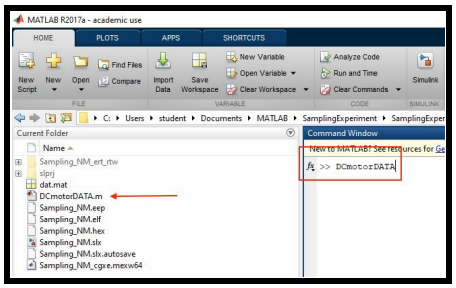

Figure F.27: AutoScaled Correct Data

68. Disconnect the power supply from the Arduino.

69. Once the plot is stopped, the variables shown in Figure F.28a should now be in the Matlab workspace. The “WindowData" variable is the data from the serial plot window. Type *DCMotorDATA* in the Matlab command window as shown in Figure F.28b. This will run the “DCMotorDATA.m" file that was inside the “SamplingExperiment" folder that was downloaded in the Software Setup section. This file will produce two figures, the motor position in counts, which is the “WindowData" variable, and the velocity of the motor in counts/sec. Both figures plot the data versus time, and not versus samples, as it did in the serial plot window. This will allow you to check if the frequency of the sine wave is correct.




(a) Workspace Data from Plot Serial




(b) Running the Plot File

Figure F.28: Handling the Serial Plot Data

70. Save all your figures for your lab report.

71. After you have saved your figures, in the Matlab command window, type *close all* and press enter to close all figures. Type *clear all* and press enter to clear the workspace of all variables. Then type *clc* and press enter to clear the command window.

Table F.3: Sample Intervals to Experiment With in Hardware

| Sampling Interval $T_s$ | Number of Samples |
|---|---|
| 0.0625 | 64 |
| 0.125 | 32 |
| 0.25 | 16 |

72. Now repeat the process for collecting data by setting *Ts* in the command window to the corresponding sampling intervals in Table F.3. Do not forget to also put in *rp* in the command window as *(2*pi)/64*. Then download the model to the Arduino. The number of samples corresponds to the number of samples in the serial plot.

73. Once you have completed the other two sampling rates, save the Simulink file, and disconnect the USB cable from the PC.

74. Calculte the frequency of each of sine wave from the plots. Do they match your expectations?

75. Compare the simulated sine wave plots with the experimentally collected sine wave plots. Are they similar? Why or why not?

76. What do you think is the best sampling rate? What are the disadvantages of making the sampling rate too small? What are the disadvantages of making the sampling rate too large?

### F.4 Table of Discussions and Questions

Before you turn in your report for this experiment, make sure that you have answered all of the questions that have been posed. It is important that your answers be expansive and that they demonstrate that you were mentally engaged in the experiment. Below is a recap of the important questions and the number of the step where each question was embedded.

| steps | Discussion/Question |
|---|---|
| 19 | Hand sketch continuous time sine wave |
| 20 | Hand sketches of sampled sine wave |
| 34 | Nyquist rate of sine wave |
| 37 | Plotted sine waves in Simulink |
| 38 | Comparison of hand sketch to Simulink simulations |
| 39 | Points on each plot |
| 40 | $T_s = 0.9$ |
| 72 | Experimental plots for each sampling rate |
| 74 | What frequency appears in each sine waves? |
| 75 | Comparison of simulation and experiment |

| | |
|---|---|
| 76 | Advantages of different sampling rates |

## F.5 Conclusion

This experiment was focused on the concept of sampling. You theoretically investigated sampling sine waves, and then compared that with computer simulations. Finally, you used Simulink to program the Arduino to sample the motor encoder and measure the motor velocity at several different sampling rates. If the sampling rate is too small, it is not possible to reconstruct the original continuous signal. In the remaining experiments, you will be using the Simulink models for applying a PWM signal to the motor and accessing the motor encoder to measure the motor velocity and position.

# APPENDIX G

## Open Loop Step Response Handout

### G.1 Objective

The objective is to find a first-order model for a DC motor using the open loop step response. You will be given some of the motor parameters from the motor data sheet and will derive the rest of the parameters using the motor step response. You will then compare the actual response of the motor with the theoretical response.

### G.2 Setup

#### G.2.1 Required Materials

**Hardware**

All hardware from the *Simple DC Motor* experiment will be required.

**Software**

- Matlab/Simulink 2017a
- Windows 10

**Prerequisite Experiments**

This laboratory requires that the following laboratories have been completed:

- Simple DC Motor
- Sampling and Data Acquisition

### G.2.2 Software Setup

1. Open your internet browser and navigate to `thl.okstate.edu`.

2. On the left side of the Homepage, select "Experiments"

3. In the middle section of the Experiments page select “All"

4. In the middle section of the All Experiments page find and select *Open Loop Step Response*.

5. On the Open Loop Step Response page select “Software/Code" in the rightmost section. Download the zipfile named *Software_Files*.

6. Right-click the file and choose “Extract All...", or use any other method of extracting the files on your PC. **Note:** Extract the software to the path to *C:\Users\student\Document\MATLAB* so that the function is in your main MATLAB path. Also replace *student* to match your path.

## G.3 Experimental Procedures

### G.3.1 Exercise 1: Deriving Motor Transfer Function

Figure G.3.1 below is the circuit diagram for the DC motor:

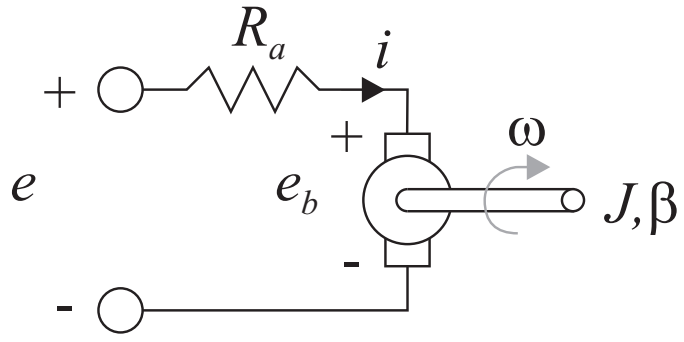



Figure G.1: Circuit Diagram for DC Motor

The equations of motion for the system are:

$$e = R_a i + e_b \tag{G.1}$$

$$J\dot{\omega} = \tau - \beta\omega \tag{G.2}$$

$$\tau = K_t i \tag{G.3}$$

$$e_b = K_b \omega \tag{G.4}$$

where $J$ is the inertia of the armature and the load, $\beta$ is the viscous friction coefficient, $K_t$ is the torque constant, $R_a$ is the armature resistance, $K_b$ is the back emf constant, $\tau$ is the torque, $i$ is the armature current, $e$ is the voltage applied to the motor, $\omega$ is the angular velocity of the motor, and $e_b$ is the motor back emf.

7. Solve for the transfer functions $G_1, G_2, G_3,$ and $G_4$ in Figure G.2, using Equations (G.1) - (G.4) .

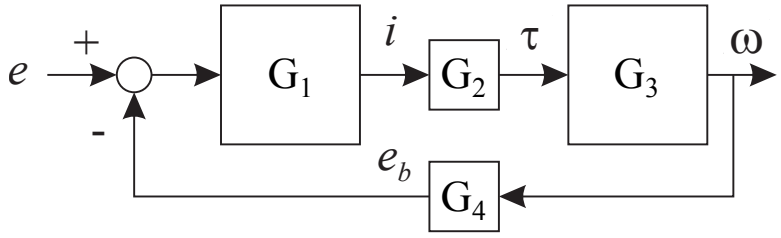



Figure G.2: Empty Block Diagram for Motor

8. Using the following block diagram reduction equation, find the overall open loop transfer function $\frac{\Omega(s)}{E(s)}$.

$$G(s) = \frac{\Omega(s)}{E(s)} = \frac{G_1(s)G_2(s)G_3(s)}{1 + G_1(s)G_2(s)G_3(s)G_4(s)} \tag{G.5}$$

### G.3.2 Exercise 2: Theoretical Open Loop Step Response

In this exercise you will derive the theoretical open loop step response for the DC motor. In later exercises, you will compare this **theoretical** response to **experimental** and **simulated** step responses.

9. Arrange your open loop transfer function $G(s) = \frac{\Omega(s)}{E(s)}$ from the previous step in the form

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_m}{\tau_m s + 1} \tag{G.6}$$

10. $K_m$ is the open loop DC gain, and $\tau_m$ is the time constant. Assume that a step input of $A$ volts is applied to the motor with the initial motor velocity equal to zero. Derive the motor velocity $\omega(t)$ using partial fraction expansions, and plot the velocity versus time. Your step response should be a function of $K_m$ and $\tau_m$.

11. How does $K_m$ affect the step response? How does $\tau_m$ affect the step response? In a later exercise you will find the step response of your DC motor experimentally, and you will use the step response to determine $K_m$ and $\tau_m$.

12. Find and plot the pole of your open loop transfer function. As the pole moves to the left in the complex plane, how would the system time response change?

### G.3.3 Exercise 3: Open Loop Step Response Variables

To produce the open loop step response (both experimentally and in simulation), you will need to create a Matlab file (used to store variables into the Matlab workspace) before running Simulink. Follow the steps below to create this Matlab file.

**Setting Up Matlab File**

13. Open Matlab 2017a. In the top left-hand corner of the main Matlab page, click the button labeled “New Script" (See Figure G.3).

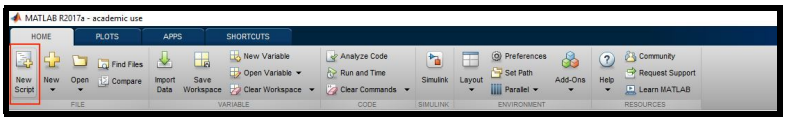



Figure G.3: New Script Button Location on Main Matlab Page

14. A page labeled “Editor-Untitled" should be visible. This is the Matlab “m-file" editor. Save the m-file by clicking the **Save** button  at the top left hand corner.

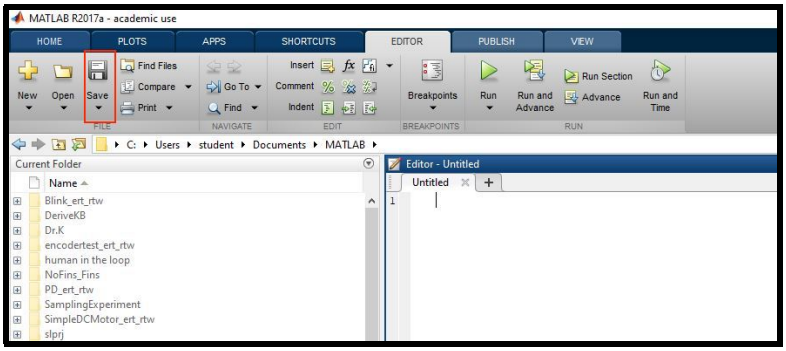



Figure G.4: Save Button Location on m-file Editor Page

15. A page labeled “Select File for Save As" should now be visible. Navigate the browser to the directory where your previous experiments were saved. Type **OL_Constants.m** as the File name. Save the file in this directory by clicking **Save** at the bottom.

16. Type **Ts = 0.01;** (See Figure G.5). This will be the sampling time.

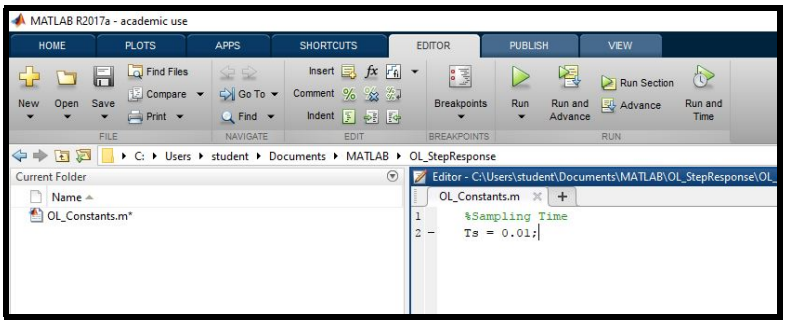



Figure G.5: Sampling Time Variable “Ts" Added to m-file

17. Type **rp = 2*pi/64;** the Pololu motor/encoder has a resolution of 64 counts per revolution. This means that every time the motor has completed a full revolution ($2\pi$ radians), the encoder has provided 64 counts. Multiplying the output of the encoder count by **rp** produces the angular position of the motor shaft in radians.

18. Type **A = 3;**, **T=60;**, and **D = 50;** as shown in figure G.6. This will be used in the next section to create a sequence of pulse voltage inputs to the motor, with amplitude 3 volts, period 6 sec and 50% duty cycle.

```
1        %Sampling Time
2 -      Ts = 0.01;
3
4        %Encoder Resolution
5 -      rp = 2*pi/64;
6
7        %Amplitude
8 -      A = 3;
9
10       %Pulse Period
11 -     T = 60;
12
13       %Pulse Width
14 -     D = 50;
```

Figure G.6: Final Matlab Variables

19. Save the file again by clicking **Save**  at the top of the page. This completes the Matlab file setup.

### G.3.4 Exercise 4: Experimental Open Loop Step Response

In this exercise, you will create an experimental Simulink model to load to the Arduino. We call it experimental because it will be loaded to the Arduino's memory and run to collect data from the open loop step response of the motor. This open loop step response data will then be interpreted in the next exercises to find the appropriate $K_m$

and $\tau_m$ values.

**Setting Up Simulink File (Arduino)**

20. Open the Simulink model that was created in the *Sampling and Data Acquisition* laboratory, as shown in G.7. Save this file as **OL_Step_Resp_Arduino.slx** to the same folder where **OL_Constants.m** is saved

21. In the Simulink model, delete the sine block, and replace it with a **Pulse Generator** block, and connect to the **Abs** block and to the middle port of the **Switch** block.

22. Double-click on the **Pulse Generator** block and change the “Amplitude value:" to **A**. Change the "Period (secs) value:" to **T**. Lastly change the "Pulse Width (% of period) value:" to **D**. The pulse generator will generate a series of pulses of amplitude **A**. The pulses will occur every **T** seconds and will last for **D** % of the period **T**. This will produce a series of step responses of the motor.

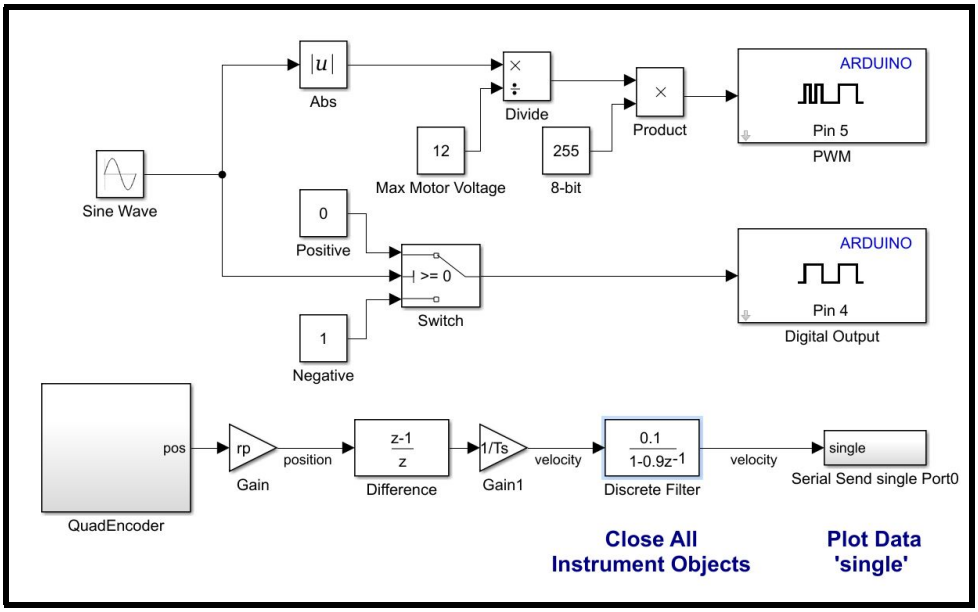



Figure G.7: Simulink Model from Sampling and Data Acquisition

23. Now the Simulink file that will be deployed to the Arduino is complete (see Figure G.8 for the final Simulink model). Before loading this model to the Arduino,

ensure that the Arduino is connected to your computer and the power supply is connected to the motor shield. Save this file as **OL_Step_Resp_Arduino.slx** to the same folder where **OL_Constants.m** is saved.

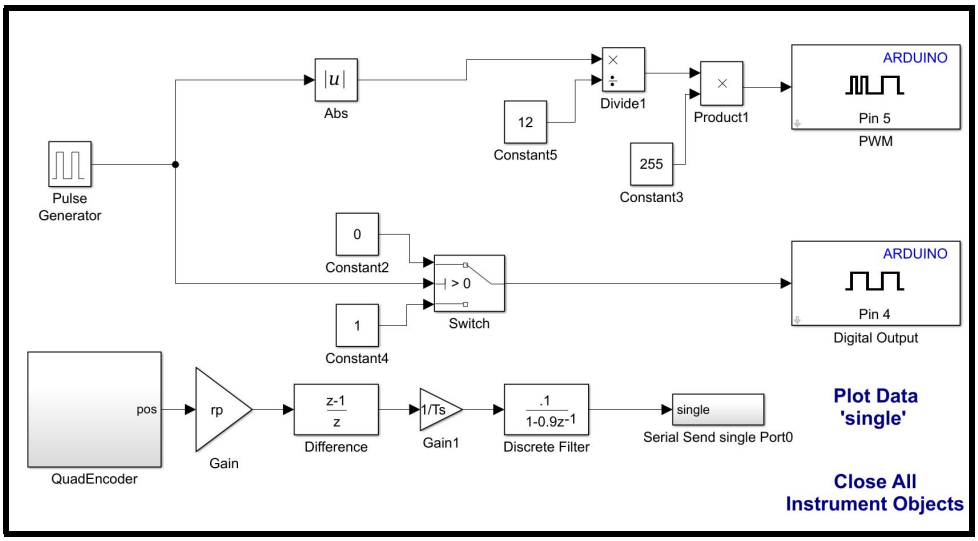



Figure G.8: Final Simulink Diagram to Load on Arduino

### Collecting Experimental Data

The steps below will explain the common method for capturing data from the Arduino. If you run into issues, refer back to the *Sampling and Data Acquisition* experiment.

24. Open **OL_Constants.m** and click the Run button  at the top of the page.

25. Open **OL_Step_Resp_Arduino.slx** and click the “Deploy to Hardware" button  at the top-right of the page.

26. Once the model has successfully deployed to the Arduino, double click on the text “Plot Data ‘single‘" inside the model window.

27. When the small window labeled “Plot Ser..." appears, enter the Arduino COM port number under “Enter COM port to collect data:." Also, under “Enter Data Type:" type **single**, and for “Enter Number of samples to plot:" type **12000**.

**NOTE:** To find the COM port number for your Arduino, refer to the *Simple DC Motor* experiment under the section “Software Setup → Installing Arduino Mega 2560 Drivers."

28. Click Okay. Once the plot appears, plug the power cord from the power supply into the motor shield.

    **CAUTION:** Do not put your hands or any other parts of your body in front of the motor load’s trajectory. Also, if the load does not begin spinning once the motor shield is plugged in, immediately unplug the power and check to see if everything is connected properly (review the “Hardware Setup" section in the *Simple DC Motor* for the proper hardware connections.)

29. Observe the plot. If the plot appears to be very jumpy (meaning that the values do not look to be “smooth" and vary from extremely positive to negative values) or if the values are not changing from zero, proceed to the next step. If the plot appears to be smooth (rising to a value and staying around that value, as in your theoretical plot of Exercise 3, and then later dropping back to zero) then **skip to step 33** of this section; otherwise, **continue to the next step**.

30. **Unplug the power** from the motor shield, ensuring the motor load has come to a complete stop. **Press the reset button**, which should be located under the motor load trajectory on the Arduino.

31. Once the Arduino has fully reset, the plotting window should appear to output a value of zero (flat line). If this is not the case, press reset once more until the plot displays zero.

32. Plug the power cable back into the motor shield. If the data appears to be increasing smoothly (not varying to very large and small values), **continue to the next step**. If the data is still not smooth and largely varying, **repeat the above two**

**steps**.

33. Let the data fill the plot window as it moves to the left in the plot window. Once the data (starting with a value of zero) has filled the plot window completely, click the “Stop" button at the bottom of the screen. You should now have 6000 data points inside the window (which is 60 seconds.)

    **NOTE:** You should expect to see a velocity curve beginning at zero, rising up and settling to a constant value, as in your theoretical plot from Exercise 3, and then dropping back to zero as each pulse input ends. Make sure you have at least one full pulse inside the window.

34. In the upper left-hand corner of the figure click “File" and click “Save As."

35. Save the figure as **OL_data.fig** in the same folder as all of the other files you have created in this experiment and click Okay. Close the figure. Another “Plot Ser..." window should appear. Click the close button  on the window.

36. Navigate over to the main Matlab window. Under the Workspace, find the variable **WindowDat**, right-click it and click “Save As". Name the file **OL_data.mat** and save it into the same folder where you saved **OL_data.fig**.

37. You now have the experimental data for the open loop step response of this experiment.

### G.3.5 Exercise 5: Find Transfer Function from Experimental Data

**Experimental Plotting File**

38. Open the main Matlab 2017a window and click **New** at the top and then click **Script**.

39. Once the new Untitled m-file appears, Click **Save**  at the top of the page. Save the file as **OL_Plot.m**.

40. Copy and paste the text in Listing I.1 into the Matlab file. After adding the code click **Save**  and then click **Run**  . . **Note**: some things might not copy and paste correctly, so you may need to edit the script to look like Listing G.1 (e.g., the minus sign might not get converted correctly, and the last few lines might get pasted onto one line).

Listing G.1: Code for Plotting Experimental Results

```
%Load the Experimental data and store into a variable
Exp_dat = load('OL_data.mat');
Exp_dat = Exp_dat.WindowDat;

%Function for lining up step response
[y_pulse,~,~,~] = findShift2(Exp_dat,T*(1/Ts),D*T,A);

%set the time vector to be the same length as y_pulse (30 seconds)
time = Ts*(0:(length(y_pulse)-1));

%Plot the Experimental Data with Respect to Time
figure;
plot(time,y_pulse,'Color','r','LineWidth',1);
title('Experimental Open Loop Step Response'); %Title
legend('Experimental','Location','northwest'); %Legend
xlabel('Time(seconds)'); %x-axis Label
ylabel('w(radians/second)'); %y-axis Label
```

41. Compare the angular velocity vs. time plot found in the previous step to your theoretical plot from Exercise 3. Calculate the time constant ($\tau_m$).

42. Calculate the DC gain ($K_m$) from the experimental plot. (Remember that, for this experiment, the step magnitude was $A = 3$.)

43. Using the calculated $\tau_m$ and $K_m$, find the first order transfer function. Use the following form for the transfer function:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_m}{\tau_m s + 1} \tag{G.7}$$

44. Compare the transfer function found in Equation G.7 with the transfer function found using the block diagram reduction in Exercise 1. How are the transfer functions related? Find expressions for $K_m$ and $\tau_m$ in terms of $K_t$, $K_b$, J, $\beta$ and $R_a$.

45. Given that $K_t = 0.0058\ \frac{Nm}{A}$, $K_b = 0.0058\ \frac{\frac{V}{rad}}{s}$ and $R_a = 2.6\ \Omega$, calculate the remaining motor parameters ($\beta$ and J).

46. You may wish to rerun the experiment to see if you get consistent values for $K_m$ and $\tau_m$. If the results are not consistent, explain why that could be.

### G.3.6 Exercise 6: Compare Simulation and Experimental Results

Typical engineering practices involve simulation followed by experimental testing on a real system. For this experiment, the experimental values are obtained first in order to find the model parameters for simulation. Now that those values have been found, a Simulink model for simulation will be set up. The simulated results will then be compared with the experimental results by overlaying the plots. The setup of the Simulink simulation file will be explained, and then you will be given steps to take the data from the simulation and plot them in the same figure as the experimental results.

#### Setting Up Simulink File (Simulation)

47. With MATLAB R2017a open, click the button labeled “New" to reveal the drop-down menu. Under the blue bar labeled “Simulink" click the button labeled “Simulink

Model".

48. With a new Simulink Model labeled “untitled" visible, click on “File" at the top left hand corner of the page. In the drop-down menu, click “Save As...."

49. In the “Save As" page, navigate to the folder where you have saved the Matlab file created from section **Setting Up Matlab File**. Name the file **OL_Simulation.slx** and click **Save.**

50. Click the **Model Configuration Parameters** icon  at the top of the page.

51. On the left column of the page click **Solver** (“Simulation time" should appear at the top of the window.)

52. Change the “Stop time:" to **30**.

53. Next to “Type:" click the drop down menu and choose **Fixed-step**. Once available, change the value for “Fixed-step size (fundamental sample time):" to **Ts** and then click “Okay."

54. In the Simulink model, open the Library Browser and place a **Constants** block. Double-click on the Constants block and change the value to **3**. Change the name of the block from “Constants" to “Input Voltage".

55. Add a **Transfer Function** block to the Simulink model. Double-click on the block, add the value **[Km]** to the box labeled “Numerator coefficients", add the value **[tau 1]** to the box labeled “Denominator coefficients" and then click OK (See Figure G.9).

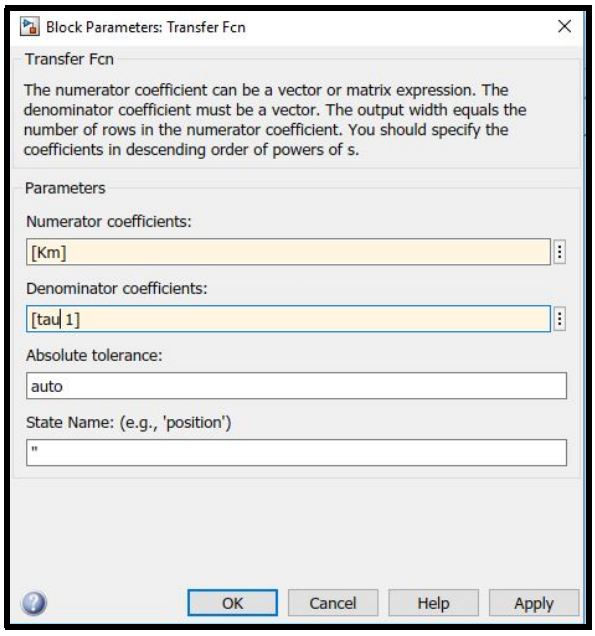



Figure G.9: Function Block Parameters

56. Add a **Scope** block to the Simulink model. Connect the input of the Scope block to the output of the Transfer Function block.

57. Add a **To Workspace** block to the Simulink model. Connect the output of the Transfer Function block to the input of the **To Workspace** block.

58. Double-click on the **To Workspace** block and type **velocity** in the box labeled “Variable name:" and then click OK.

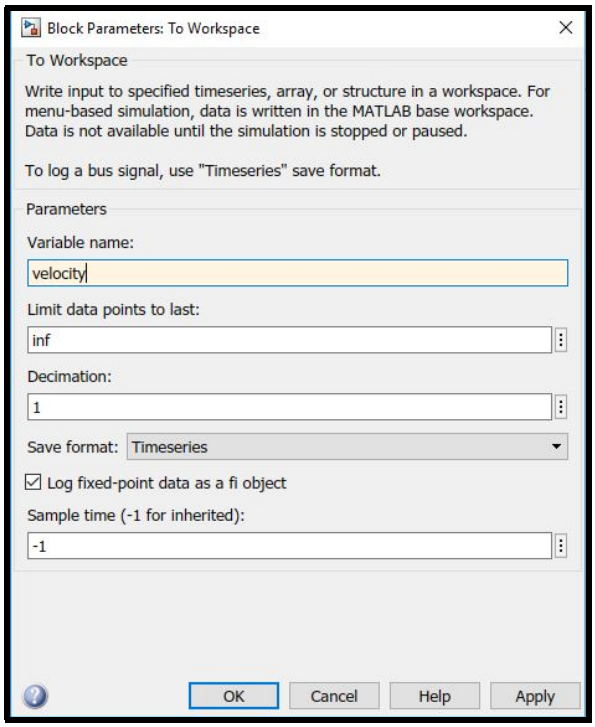



Figure G.10: Changing Variable Name in Workspace Block

59. The Simulink simulation model is now complete (see Figure G.11).

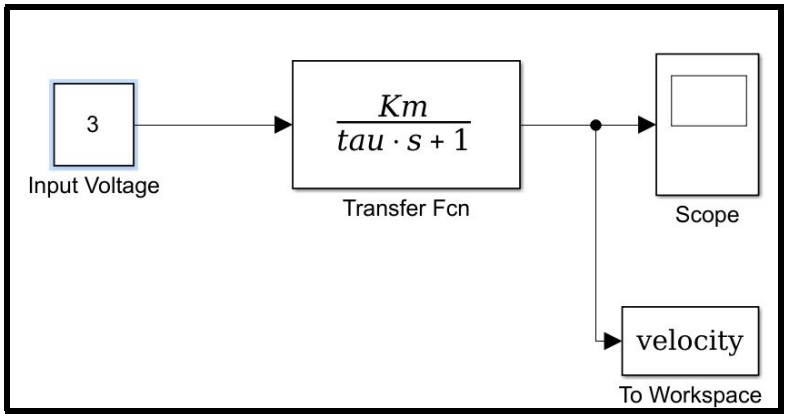



Figure G.11: Final Simulink Diagram to be Simulated and Compared With the Experimental Results

### Collecting Simulation Data

60. Open the **OL_Constants.m** file created in Exericise 1.

61. Add the lines of code found in Figure G.12 to the bottom of the m-file.

```
16      %Observed Km and tau from experimental results
17 -    Km = 0; %*** REPLACE 0 WITH Km VALUE
18 -    tau = 0; %*** REPLACE 0 WITH tau VALUE
```

Figure G.12: Add These Two Lines to Code

62. Replace each **0** value in the **OL_Constants.m** file (shown in Figure G.12) with the values you found for $K_m$ and $\tau_m$ in steps 41 and 42.

63. Save the file and then press the “Run" button  at the top of the page. Navigate to the “MATLAB 2017a" home page. Under “Workspace" on the right-hand side of the page, all of the variables from “OL_Constants.m" should be listed below.

    **NOTE:** if any variables created in the “Setting Up Matlab File" section are not listed under “Workspace," simply add the missing variables to the bottom of the **OL_Constants.m** file and click **Save** once more. If any variables are missing from the OL_Constants.m file, the “OL_Simulation.slx" file will not run correctly.

64. Open **OL_Simulation.slx**. Click the Run button  at the top of the page.

65. Once the model has finished running, double-click on the **Scope** block. Click the **Autoscale** button  .

    **NOTE:** Observe the plot; does the response look similar to the response found in step 33? If not, check the $K_m$ and $\tau_m$ values you found in steps 41 and 42. For now, the exact $K_m$ and $\tau_m$ values do not matter, just the general shape of the plot (which should be increasing until it reaches a steady state value, where it stays constant, as in the theoretical plot you made in Exercise 3).

66. Navigate back to the “MATLAB 2017a" home page. Under “Workspace" a new variable **velocity** should now be available. Right click on **velocity** and click “Save As..."

Navigate to the folder in which you have saved this project, name the “File name:" as **CLSR_1.mat**, and click **Save**  at the bottom of the page.

**Experimental vs. Simulation Comparison Matlab File**

67. Open the **OL_Plot.m** file you created in steps 38 - 40.

68. Copy and paste the text from Listing G.2 to the bottom of **OL_Plot.m**. Click **Save**  .

Listing G.2: Code for Plotting Simulation Results (Added to **OL_Plot.m**)

```
%Load the simulated data and store into a variable
Sim_dat = load('CLSR_1.mat');
T1 = Sim_dat.velocity.Time;
Sim_dat = Sim_dat.velocity.Data;

%Plot the Simulated Data with Respect to Time
hold on;
plot(T1,Sim_dat,'Color', 'k', 'LineWidth', 2);
legend('Experimental', 'Simulation', 'Location', 'northwest');
```

69. Click **Run**  at the top of the page.

70. Compare the responses of the simulation with the experimental data. Does the simulated response match the experimental response? Describe the similarities and differences. Write down the current value for $K_m$ and $\tau_m$ on a piece of paper to keep track of them.

71. What would cause the differences between the responses?

72. If the plots look different, change the values for $K_m$ and $\tau_m$ to match the simulation with the experimental open loop step response. After finding the new values, change the $K_m$ and $\tau_m$ values in the **OL_Constants.m** file. After changing the

**OL_Constants.m** file, follow steps 60 - 69 once more. If the simulated and experimental open loop step responses look almost the same, **continue to the next step**.

73. How much (if any) did the new values for $K_m$ and tau change from your original values? For the new values of $K_m$ and $\tau_m$, repeat step 45. How much did the values for J and $\beta$ change?

74. Finally, compute the pole of your system based on the transfer function you have derived.

### G.4 Table of Discussions and Questions

Before you turn in your report for this experiment, make sure that you have answered all of the questions that have been posed. It is important that your answers be expansive and that they demonstrate that you were mentally engaged in the experiment. Below is a recap of the important questions and the number of the step where each question was embedded.

| steps | Discussion/Question |
|---|---|
| 7 | Solve for $G_1, G_2, G_3, G_4$ |
| 8 | Find $\frac{\Omega(s)}{E(s)}$ |
| 9 | Arrange in the form $\frac{K_m}{\tau_m s+1}$ |
| 10 | Derive and plot response of $\omega(t)$ |
| 11 | How does $K_m$ and $\tau_m$ affect the step response? |
| 12 | Plot of pole along with description. |
| 41 | Comparison of angular velocity vs. time of theoretical and experimental |
| 41 | Calculate time constant $\tau_m$ |
| 42 | Calculate the DC gain $K_m$ |
| 43 | Finding first order transfer function with calculated experimental values |

| | |
|---|---|
| 44 | How does the theoretical transfer function relate to the experimental transfer function? |
| 44 | Find expressions for $K_m$ and $\tau_m$ in terms of $K_t, K_b, J, \beta$, and $R_a$ |
| 45 | Calculate $\beta$ and J |
| 46 | Explanation of why the results may not be consistent |
| 65 | Does the response look similar? |
| 70 | Comparison of experimental and simulated response |
| 71 | What could cause the differences between the responses |
| 72 | Final plots |
| 73 | Changes from initial to final values of $K_m, \tau_m, \beta$, and J |

## G.5 Conclusion

This experiment provided an analysis of the open loop step response of a DC motor. Theoretical, experimental and simulated responses were generated and compared. This experiment utilized the *Simple DC Motor*and *Sampling and Data Acquisition* experiments.

# APPENDIX H

## Closed Loop Step Response Handout

### H.1 Objective

This experiment adds feedback to the *Open Loop Step Response* experiment. The objective is to investigate how feedback changes the system poles and how changes in the pole locations affect the time response of the system. You will estimate such characteristics as settling time, percent overshoot and steady state error. You will verify your calculations through simulations and experiments. Using the Arduino, you will implement a proportional and derivative controller and will find the system response to a step change in the reference motor position. By comparing the theoretical, simulated and experimental responses you will discover the strengths and limitations of linear modeling.

### H.2 Setup

#### H.2.1 Required Materials

**Hardware**

All of the materials used in the *Open Loop Step Response* experiment will be used in this experiment. Additionally, the hardware listed below and shown in Figure E.2 will be used.

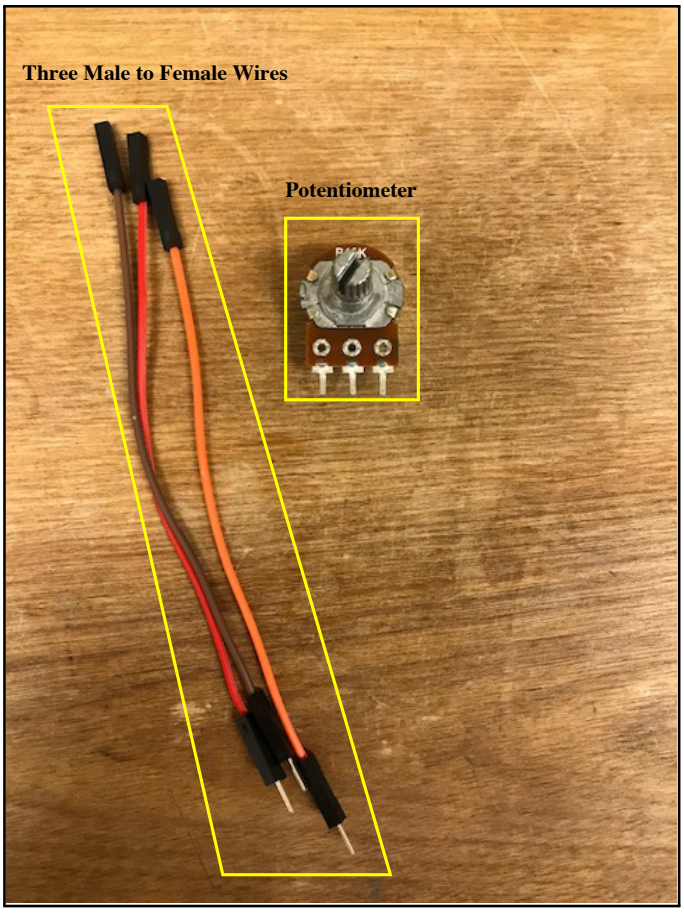



Figure H.1: Hardware Required for Laboratory

- Potentiometer
- 3 Male-to-Female wires

### Software

- All software from the *Open Loop Step Response* experiment is required for this lab.

### Previous Experiments

- *Open Loop Step Response*

### H.2.2 Hardware Setup

1. Take the potentiometer as shown in Figure H.2 and connect the female ends of three male-to-female wires to the three pins of the potentiometer, which we label as pins 1, 2 and 3 from left to right. Use sticky-tack to hold the wires in place.

2. Then take the potentiometer and insert the wire from pin 1 to GND on the Arduino, from pin 2 to Analog In 4 on the Arduino, and pin 3 to 5V on the Arduino, as shown in Figure H.3. **Note:** DO NOT WIRE THE MIDDLE PIN TO POWER OR GROUND.

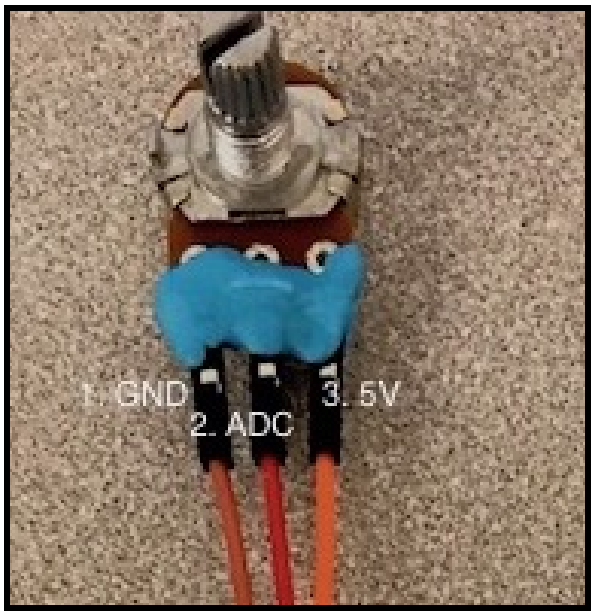



Figure H.2: Potentiometer

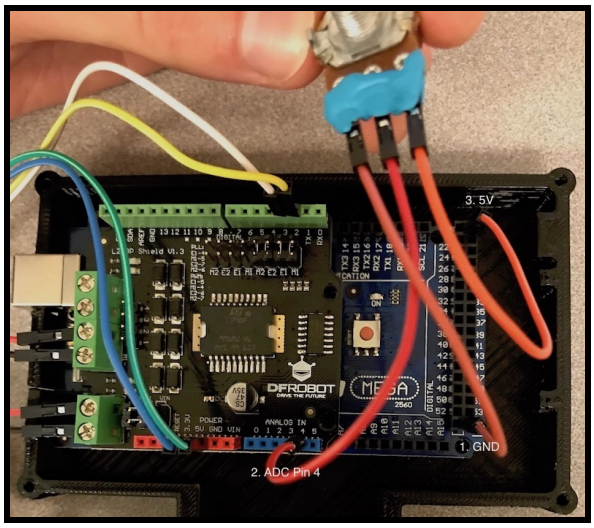



Figure H.3: Potentiometer Connected to Arduino

### H.2.3 Software Setup

No software setup is required. You should have completed the software setup in the *Open Loop Step Response* experiment.

## H.3 Experimental Procedures

### H.3.1 Exercise 1: Theory - Deriving Closed Loop Motor Transfer Function

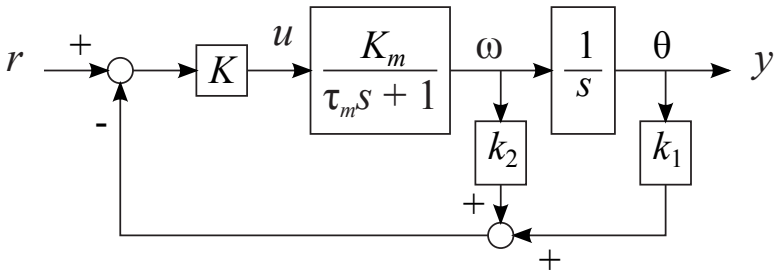



Figure H.4: Empty Block Diagram for Closed Loop Motor

3. Using block diagram reduction on the block diagram in Figure H.4, find the closed loop transfer function $\frac{\mathrm{Y}(s)}{\mathrm{R}(s)}$.

4. Assume that the reference input, $r(t)$, is a step function of amplitude $\frac{\pi}{2}$. You calculated $\mathrm{K}_m$ and $\tau_m$ from the *Open Loop Step Response* experiment. Consider the

following two sets of gains:

- K = 3, $K_2$ = 0.0, $K_1$ = 1
- K = 3, $K_2$ = 0.1, $K_1$ = 1

First, find the closed loop poles for both sets of gains. Based on the pole locations, estimate the settling time, percent overshoot, and frequency of oscillation of the closed loop step response for each set of gains. Also, find the steady state value from the closed loop transfer function.

5. Using the values you computed in the previous step, hand sketch the closed loop step responses for each set of gains. (You will make two different plots, one for each set of gains.) Show scales on all axes.

### H.3.2 Exercise 2: Simulation - Simulated Closed Loop Step Response (First Feedback Gain Set)

You will first simulate the closed loop step response before finding the step response experimentally. This exercise will provide steps for editing the **OL_Simulation.slx** file you created in the *Open Loop Step Response* laboratory to produce a simulated closed loop step response.

**Setting Up Matlab File**

6. Open the **OL_Constants.m** file created in the open loop step response experiment.

7. Save the file as **CL_Constants.m.**

8. Delete **A=3;** and type **ref=pi/2;**. This will be the reference position for the closed loop step response.

9. Change **T=60;** to **T=20;**

10. On the next line type **K = 3;** This will be the gain that is multiplied by $r-(k_2\omega+k_1\theta)$. Since $k_1$ is always equal to 1, K effectively multiplies $(r-\theta)-k_2\dot{\theta} = e-k_2\dot{\theta}$. The term $Ke$ is called proportional feedback, since it produces an input that is proportional to the error. The term $Kk_2\dot{\theta}$ is the derivative feedback, and has a damping effect, like viscous friction.

    **NOTE:** $k_1$ will not be added to this Matlab file, since it will always be equal to 1 for this experiment.

11. Type **K2 = 0.0;**

    **NOTE:** The gains K and $k_2$ (given in Step 2) will be reassigned with new values corresponding to the second set of gains in Exercises 4 and 5.

```
%Sampling Time
Ts = 0.01;

%Encoder Resolution
rp = 2*pi/64;

%Reference value
ref = pi/2;

%Pulse Period
T = 20;

%Pulse Width
D = 50;

%Observed Km and tau from Open Loop Step Response experiment
Km = 0.0;   %Replace 0 with Km value from previous experiment
tau = 0.0;  %Replace 0 with tau value from previous experiment

%Gains
K = 3;    %Proportional gain
K2 = 0.0; %Derivative gain
```

Figure H.5: Matlab File for Closed Loop Experiment

**Setting Up Simulink File (Simulation)**

12. Open the **OL_Simulation.slx** file created in the *Open Loop Step Response* experiment.

13. Save the file as **CL_Simulation.slx.**

14. Change the simulation time to 10 seconds.

    **NOTE:** You will run the file for 10 seconds instead of 30 seconds because the closed loop step response will have a shorter settling time than the open loop step response.

15. Delete the connections from the **Transfer Function** block to the **Scope** and **To Workspace** blocks.

16. Additionally, delete the connection between the **Input Voltage** and the **Transfer Function** blocks.

17. Add 2 **Sum** blocks, 2 **Gain** blocks, and 1 **Integrator** block to the Simulink model.

18. Rename the **Input Voltage** block to **Reference.** Double-click on the same block and change the “Constant value:" to **ref.**

19. Rename one of the **Gain** blocks to **K.** Double-click and change the “Gain:" value to **K.**

20. Rename the other **Gain** block to **K2.** Change the **Gain:** value to **K2.**

21. Connect the output of the **Reference** block to the input of one of the **Sum** blocks. Connect the output of the **Sum** block to the input of the **K** block. Connect the output of the **K** block to the input of the **Transfer Function** block.

22. Double-click on the **Sum** block from the previous step and next to “List of signs:" change the second + sign (farthest to the right) to a **-** sign and click okay.

23. Connect the output of the **Transfer Function** block to the input of **Integrator** block. Connect the output of the **Integrator** block to the input of one of the **Scope** blocks. Rename the same **Scope** block to **Angular Position.**

24. Right-click on **K2** and hover the mouse over **Rotate & Flip.** Left-click on **Clockwise.**

25. Connect the input of **K2** to the connection between the **Transfer Function** and the **Integrator**.

26. Double-click on the unused **Sum** block. Next to “List of signs:" move the two **+** signs to the right of the | sign (See Figure H.6)

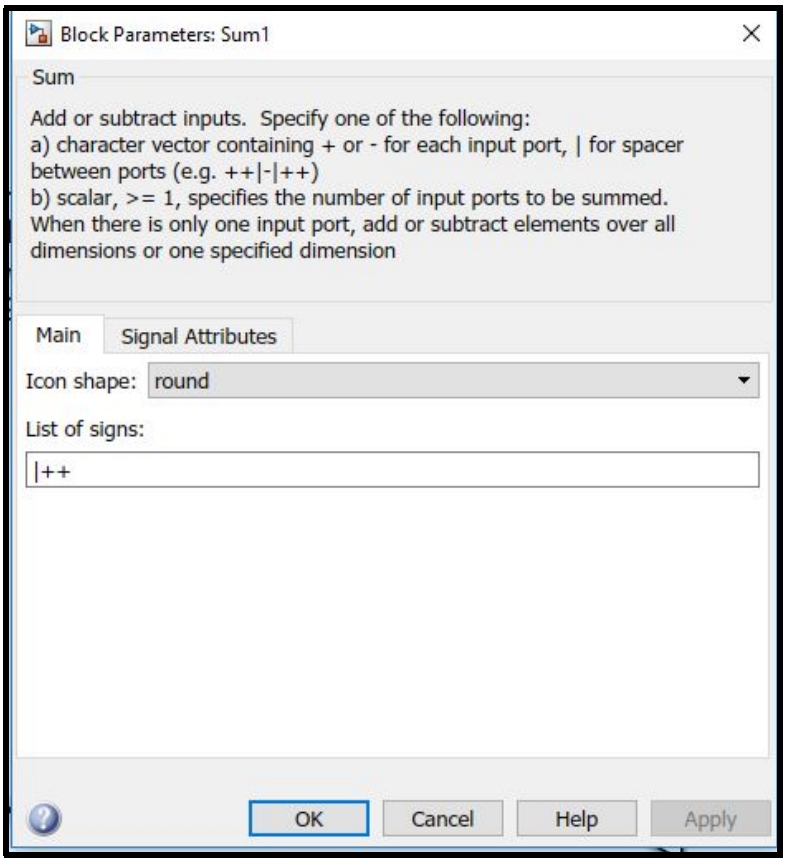



Figure H.6: Correct Setup for Second Sum Block

27. While the **Sum** block is still highlighted (the block is outlined in blue) hold down the key combination **Ctrl-R** to rotate the block Clockwise and then let go. Once again, hold down **Ctrl-R** and let go of the keys to rotate it clockwise one more time. The **Sum** block should now have two inputs: one facing up and the other facing right. The output should be facing to the left.

28. Connect the output of **K2** to the input (facing in the upward direction) of the **Sum** block from the previous step. Connect the input (facing to the right) of the **Sum** block to the output of the **Integrator** block and the input of the **Angular Position** scope block.

29. Connect the output of the **Sum** block from the previous step to the **-** input of the other **Sum** block.

30. Double-click on the **To Workspace** block and type **position** in the box labeled “Variable name:" and then click OK. Connect the input of the **To Workspace** block to the connection between the output of the **Integrator** block and the input of the **Angular Position** scope block.

31. This concludes the setup of the simulation file for the closed loop step response of the motor. See Figure H.7 for the final Simulink simulation file.

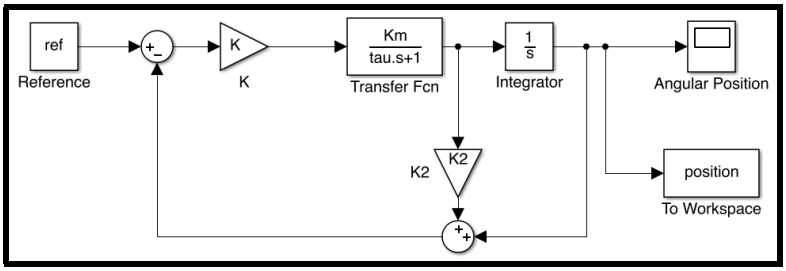



Figure H.7: Final Simulink Simulation Model Used in the Closed Loop Step Response Experiment

**Collecting Simulation Data**

32. Open **CL_Constants.m** and then press the **Run** button  at the top of the page. Navigate to the **MATLAB** command window. Under “Workspace" on the right-hand side of the page, all of the variables from **CL_Constants.m** should be listed.

    **NOTE:** if any variables created in the **Setting Up Matlab File** section are not listed under “Workspace," simply add the missing variables to the bottom of the **CL_Constants.m** and click Save once more. If any variables are missing from the file, the **CL_Simulation.slx** file will not run correctly.

33. Open **CL_Simulation.slx.** Click the Run button  at the top of the page.

34. Once the model has finished running, double-click on the **Angular Position** scope block. Click the **Autoscale** button  . Observe the plot. Does the closed loop step

response appear to rise up from zero and settle to the reference value (as in your theoretical sketch in Exercise 1)? If the plot looks to be correct (with a run time of 10 seconds) continue to the next step. Otherwise, go back to the **Setting Up Simulink File (Simulation)** section to ensure your Simulink file is correct.

35. Navigate back to the **MATLAB** command window. Under “Workspace" a variable (**position**) should now be available. Right click on **position** and click “Save As..." Navigate to the folder in which you have saved this project, type next to “File name:" **CL_position_1.mat**, and click “Save" at the bottom of the page.

36. You now have the simulation data found from Simulink for the first set of gains.

**Simulation Plot File**

37. Open the main Matlab 2017a window and click **New** at the top and then click **Script**.

38. Once the new Untitled m-file appears, Click **Save**  at the top of the page. Save the file as **CL_Plot.m**.

39. Copy and paste the text in Listing H.1 into the Matlab file. After adding the code click **Save**  and then click **Run**  .

40. Save the figure as **CL_S_1.fig** into your folder for this project. Refer to this figure for the remaining steps in this section.

41. Compare the simulation results with the hand-written results you found in Exercise 1. Does the simulated plot resemble the hand-written plot found from the first set of gains? Discuss similarities and explain any differences.

42. Estimate the settling time, percent overshoot, and frequency of oscillation of the closed loop step response from the simulated plot and compare with the hand-written plot.

43. Also compare the steady state values from each plot. Discuss the similarities and explain any differences.

Listing H.1: Code for Plotting the Closed Loop Step Response Simulated Results

```
%Load the simulation data and time and store into variables
CL_simResp_1 = load('CL_position.mat');
T = CL_simResp_1.position.Time;
CL_simResp_1 = CL_simResp_1.position.Data;

%Plot the simulation data with respect to time
figure
plot(T,ones(size(T))*ref, 'Color', 'r');
hold on;
plot(T,CL_simResp_1, 'Color', 'k','LineWidth', 2);
title('Simulated Closed Loop Step Response');
legend('Reference','Simulated', 'Location', 'southeast');
xlabel('Time (seconds)')
ylabel('Theta (radians)')
```

### H.3.3 Exercise 3: Motivation - Human in the Loop

In this experiment you will try to get the computer to turn the motor exactly 90 degrees and stop. This is harder than you think. To demonstrate how hard this is you will try to do it manually.

#### Software Setup

44. Open the Simulink file created in the *Open Loop Step Response* experiment named **OL_Step_Resp_Arduino.slx**.

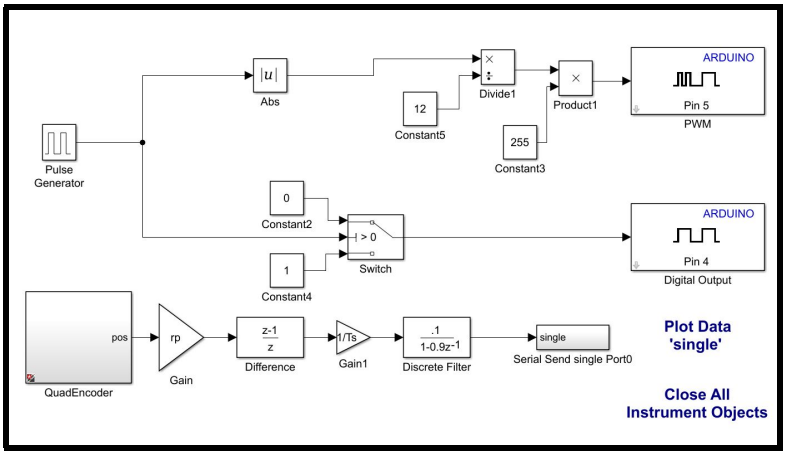



Figure H.8: Simulink Model from Open Loop Step Response Experiment

45. Add 3 **Constant** blocks, a **Sum** block, a **Divide**, a **Product**, and an **Analog Input** block.

46. Make the **Analog Input** block pin 4 and the sample time *Ts*, then connect the output of the **Analog Input** block to the input × of the **Divide** block.

47. Next, take one of the **Constant** blocks set it to *1023* and sample time to **Ts**. Then connect it to the ÷ port of the **Divide** block.

48. Take another **Constant** block, set its value to *5* and sample time to *Ts*, then connect the **Constant** block to one input of the **Product** block and then connect the output of the **Divide** block to the other input of the **Product** block.

49. Then take the last **Constant** block, set its value to *-2.5* and connect its output to the one of the inputs of the **Sum** block. Then connect the output of the **Product** block to the other input of the **Sum** block.

50. Lastly, take the output of the **Sum** block and connect it to both the input of the **Abs** block and the middle input of the **Switch** block. Refer to Figure H.9 for the final Human in the Loop configuration.

51. Plug in the Arduino and upload the model to the board.

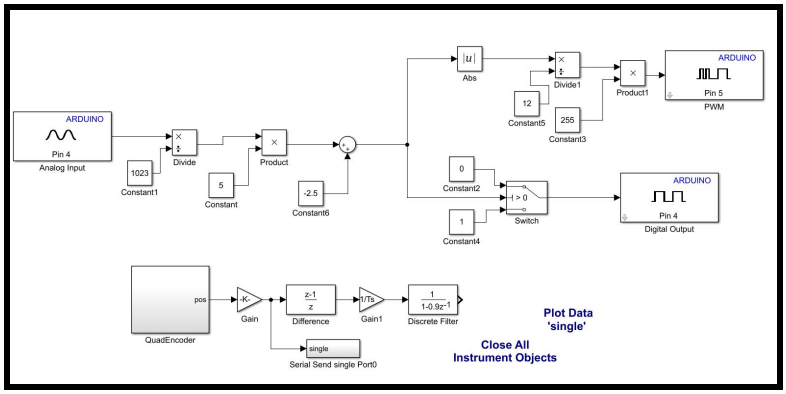



Figure H.9: Simulink Model For Human in the Loop

**Experimental Human in the Loop**

52. Plug in power to the motor shield. **Note**: Make sure nothing is in the way of the spinning load.

53. Try adjusting the voltage applied to the motor by turning the potentiometer.

54. Once you get the feel for how the potentiometer can be used to adjust the voltage to the motor and turn the load, try and get the motor to turn exactly 90 degrees and stop.

55. How difficult is it to get the motor to turn and stop at exactly 90 degrees? How accurately were you able to place the load? What is the minimum time in which you can turn the motor exactly 90 degrees and stop?

### H.3.4 Exercise 4: Experiment - Experimental Closed Loop Step Response (First Feedback Gain Set)

This section will provide the setup of the Simulink file for the Arduino.

**Setting Up Simulink File (Arduino)**

56. Again, open the Simulink file created in the *Open Loop Step Response* experiment named **OL_Step_Resp_Arduino.slx**.

57. Add 2 **Gain** blocks and 2 **Sum** blocks to the model. In the upper left hand corner of the Simulink model, click “File" and then “Save As".

    Save this file as **CL_Step_Resp_Arduino.slx** in the same folder where you saved the *CL_Constants* Simulink file **CL_Constants.m**.

58. Delete the connection from the **Pulse Generator** block going into the **Abs** and **Switch** blocks. Connect the output of the **Pulse Generator** block to the input of one of the **Sum** blocks and the output of the **Sum** block to the input of one of the **Gain** blocks. Double-click the **Sum** block and under “List of signs:" change the second **+** sign (farthest to the right) to **-**. Click Okay. The **Sum** block should have a minus sign for the bottom input.

59. Rename the **Gain** block from the previous step to **K**. Double-click and input a value of **K** for the “Gain:" value. Connect the output of the **K** block to the input of the **Abs** and **Switch** blocks.

60. Double-click on the **Pulse Generator** block and change the **Amplitude** to **ref**, and then click Okay.

61. Delete the connection from the **Discrete Filter** block to the **Serial Send single Port0** block. Connect the output of the **Discrete Filter** block to the input of the

remaining **Gain** block. Rename the **Gain** block to **K2** and double-click the block to change the “Constant value:" to **K2**.

62. Select the remaining **Sum** block and hold down the keyboard combination **Ctrl-I**. The sum block output should now be on the left and the side input should be facing to the right. Connect the output of the **K2** block into the right input of the **Sum** block. Connect the output of the **Sum** block to the **-** input of the other sum block.

63. Connect the bottom **+** input of the **Sum** block to the connection between the **Steps to Radians** block and the difference block. Connect the input of the **Serial Send single Port0** block to the connection between the **Steps to Radians** block and the difference block. The output of **Steps to Radians** should be going to the inputs of the **Difference**, **Sum**, and **Serial Send single Port0** blocks.

64. Click **Save**. The Arduino Simulink file for the experimental closed loop step response is now complete. See Figure H.10 for the completed model.

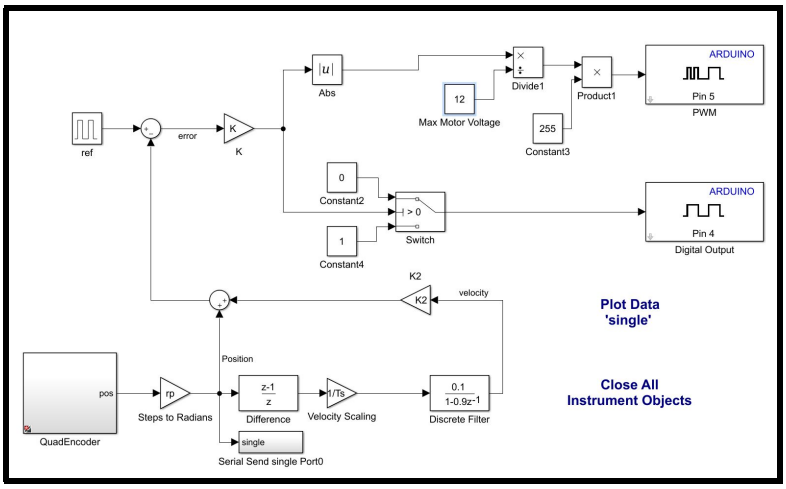



Figure H.10: Final Closed Loop Step Response Simulink Model for Arduino

### Collecting Experimental Data

65. Open **CL_Constants.m** and click the Run button  at the top of the page.

66. Open **CL_Step_Resp_Arduino.slx** and click the “Deploy to Hardware" button  at the top-right of the page.

67. Once the model has successfully deployed to the Arduino, double click on the text **Plot Data ‘single’** inside the model window.

68. When the small window labeled “Plot Ser..." appears, enter the Arduino COM port number under “Enter COM port to collect data:." The default values for “Enter Number of Samples to plot:" is “single" and for “Enter Number of samples to plot:" is **8000**. If those values are anything different, change them back to their default values.

    **Note:** To find the COM port number for your Arduino, refer to the *Simple DC Motor* experiment under the section “Software Setup → Installing Arduino Mega 2560 Drivers."

69. Click Okay. Once the plot appears, plug the power cord from the power supply into the motor shield.

    **CAUTION:** Do not put your hands or any other parts of your body in front of the motor load trajectory. If the load does not begin spinning once the motor shield is plugged in, immediately unplug the power and check to see if everything is connected properly (review the **Hardware Setup** section in the *Open Loop Step Response* experiment for the proper hardware connections.)

70. Observe the plot. If the plot appears to be very jumpy (meaning that the values do not look to be “smooth" and vary from extremely positive to negative values) or if the values are not changing from zero, proceed to the next step. If the plot appears to be smooth (rising to a value and staying around that value, as you found in Exercise 1) then skip to step 74 of this section.

71. Unplug the power from the motor shield. On the Arduino, click the reset button, which should be located under the motor load.

72. Once the Arduino has fully reset, the plotting window should appear to output a value of zero (flat line). If this is not the case, press reset once more until the plot displays zero.

73. Plug the power cable back into the motor shield, being careful not to bump the motor load (This will throw off the initial angular position of the load.) The load on the motor should move. If the data appears to be increasing smoothly, continue to the next step. If the data is still not smooth (or the initial angle is off from zero), repeat steps 71-73.

74. Let the data fill the plot window as it moves to the left. Once the data (starting with a value of zero) has filled the screen completely, click the “Stop" button at the bottom of the screen. You should now have 8000 data points on the screen (which is 80 seconds)

75. Navigate back to the **MATLAB 2017a** main page. Under “Workspace" the variable **WindowDat** should now be present. Right-click on it and click “Save As." Name the file **CL_expResp_1.mat** and save it into the folder where the **CL_Step_Resp_Arduino.slx** file is saved.

76. You now have the experimental data for the closed loop step response and the first gain set.

**Experimental Plotting File**

77. Open the **CL_Plot.m** file you created in the **Simulation Plot File** section.

78. Add the text in Listing H.2 to the bottom of the **CL_Plot.m** file. After adding the

code, click **Save**  and then click **Run**  .

Listing H.2: Code for Plotting the Closed Loop Step Response Experimental Results

```
%Load the experimental data and store into a variable
CL_expResp_1 = load('CL_expResp_1.mat');
CL_expResp_1 = CL_expResp_1.WindowDat;
[y_pulse1,~,~,~] = findShift2(CL_expResp_1,T*(1/Ts),D*T,ref);

%Plot the simulation data with respect to time
hold on;
plot(T,y_pulse1, ...
'Color', 'g', ...
'LineWidth', 1);
title('Simulated vs. Experimental Closed Loop Step Response'); %Title
legend('Reference', 'Simulated', 'Experiemental', 'Location', 'southeast'); %Legend
xlabel('Time (seconds)') % x-axis label
ylabel('Theta (radians)') % y-axis label
```

79. Save the figure as **CL_SE_1.fig** into your folder for this project. Refer to this figure for the remaining steps in this section.

80. Compare the simulation results with the experimental results you found. Do the simulated plot and the theoretical plot from Exercise 1 resemble the experimental plot found from the first set of gains? Discuss similarities and explain any differences among the three plots (theoretical, simulation, experimental).

81. Estimate the settling time, percent overshoot, and frequency of oscillation of the closed loop step response from the experimental plot. Compare with the hand-written and simulated plot.

82. Also compare the steady state values from each plot discussing similarities and explaining differences. What could cause the simulated response to differ from the experimental response?

### H.3.5 Exercise 5: Simulation - Simulated Closed Loop Step Response (Second Feedback Gain Set)

83. Open the file **CL_Constants.m**.

84. Change the K2 gain value to **K2 = 0.1;** and click **Save**  .

85. Repeat the experiment beginning with Section **Collecting Simulation Data**, steps 32-43, with the following exceptions:

    - In step 35 name the data you find from simulation as **CL_position_2.mat** for the second set of gains.
    - Change each **CL_position_1.mat** in the **CL_Plot.m** file you created in Listing H.1 to **CL_position_2.mat**.
    - Comment out the second half of the **CL_Plot.m** file pertaining to the experimental values you in Listing H.2 (for the time being, in order to observe only the Simulated Plot)
    - In step 40, save the figure as **CL_S_2.fig**.
    - In steps 41 and 42, compare **CL_S_2.fig** with the hand-written results you found for the second set of gains.

### H.3.6 Exercise 6: Experiment - Experimental Closed Loop Step Response (Second Feedback Gain Set)

86. Repeat Section **Collecting Experimental Data** steps 65-75 with the following exceptions:

    - In step 75 name the experimental data as **CL_expResp_2.mat**.
    - Before running the **CL_Plot.m** file in step 78, uncomment the second half of

the code in the **CL_Plot.m** file (Created in Listing I.2.) Change each **CL_expResp_1.mat** to **CL_expResp_2.mat** and **y_pulse1** to **y_pulse2**. After making these changes, then run the file.

- In step 79 save the figure as **CL_SE_2.fig**.
- In steps 80 - 82, compare the simulated and experimental responses in **CL_SE_2.fig** with the handwritten results you found for the second set of gains.

### H.4 Table of Discussions and Questions

Before you turn in your report for this experiment, make sure that you have answered all of the questions that have been posed. It is important that your answers be expansive and that they demonstrate that you were mentally engaged in the experiment. Below is a recap of the important questions and the number of the step where each question was embedded.

| steps | Discussion/Question |
|---|---|
| 3 | Find the transfer function $\frac{Y(s)}{R(s)}$ |
| 4 | Find the closed loop poles for both sets of gains |
| 4 | Estimate settling time, percent overshoot, and frequency of oscillation for each set of gains |
| 5 | Hand sketch the closed loop step response for each set of gains |
| 41 | Similarities and differences from hand sketch and simulation |
| 42 | Settling time, percent overshoot, and frequency of oscillation for simulation |
| 43 | Similarities and differences of steady state values of simulation |
| 55 | Speed/Accuracy of manually turning the motor exactly 90 degrees and stopping |
| 80 | Similarities and differences from theory, experiment and simulation |
| 81 | Settling time, percent overshoot, and frequency of oscillation for experiment |
| 82 | Similarities and differences of steady state values of experiment |
| 85 | Second set of gains simulation |
| 86 | Second set of gains experiment |

## H.5 Conclusion/Student Feedback

This chapter provided an experiment that adds feedback to the *Open Loop Step Response* experiment. This moved the poles of the closed loop system and produced different time responses. Simulink models for simulation were created and compared with the experimental closed loop step response results.

# APPENDIX I

## Root Locus Control Design Handout

### I.1 Objective

The objective of this experiment is to design a feedback control system for a motor positioning system. Based on the motor model you developed in the *Open Loop Step Response* experiment, you will use the root locus diagram to determine the best closed loop pole locations when using both proportional and derivative feedback. After you have simulated the response of your feedback control systems, you will test the controller experimentally. You will then iterate your design to find the best possible response, in terms of settling time, percent overshoot and steady state error.

### I.2 Setup

#### I.2.1 Required Materials

**Hardware**

- All hardware from the *Open Loop Step Response* experiment is required for this lab. (No additional hardware is required)

**Software**

- All software from the *Closed Loop Step Response* experiment is required for this lab. (No additional software is required)

**Previous Experiments**

- *Closed Loop Step Response*

### I.2.2 Hardware Setup

No hardware setup is required. You should have completed the hardware setup in the *Closed Loop Step Response* experiment.

### I.2.3 Software Setup

No software setup is required. You should have completed the software setup in the *Closed Loop Step Response* experiment.

## I.3 Experimental Procedures

### I.3.1 Exercise 1: Control Design (Proportional Feedback)

In this exercise you will design a proportional feedback controller for the DC motor, using the root locus diagram. The controller signal $u(t)$ (motor voltage) will be *proportional* to the difference between the reference signal $r(t)$ and the motor position $\theta(t)$ ($y(t)$).

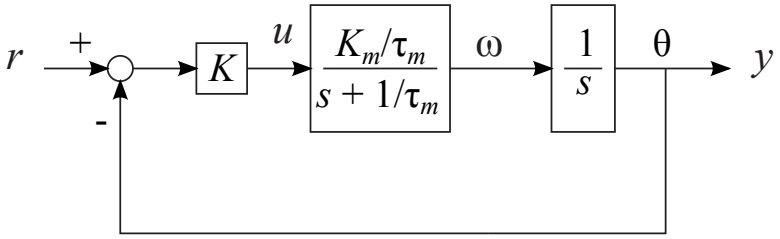



Figure I.1: Block Diagram for Closed Loop Motor with Proportional Feedback

1. Using block diagram manipulation on the block diagram in Figure I.1, find the transfer functions $G(s)$ and $H(s)$ for the equivalent block diagram in Figure I.2. Plug in the values for $K_m$ and $\tau_m$ that you found in the *Open Loop Step Response* experiment.

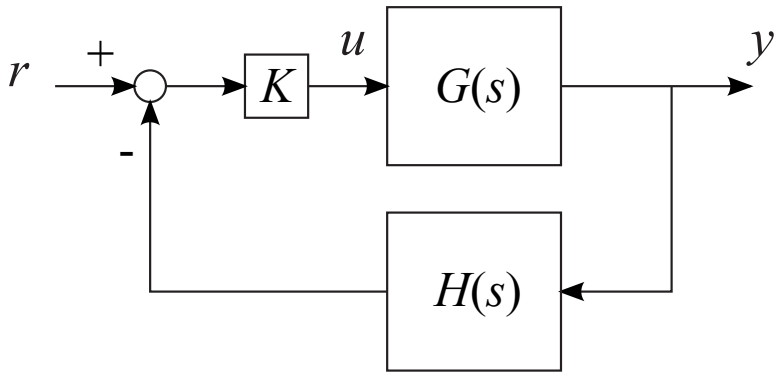



Figure I.2: Standard Feedback Control Block Diagram

2. Find the closed loop transfer function $Y(s)/R(s)$. Find the closed loop poles as a function of $K$. Complete Table I.1, computing the closed loop poles for each indicated value for $K$. In the table, P.O. is the percent overshoot of the step response, $t_p$ is the time of the first peak in the step response, and $t_s$ is the settling time (5%) of the step response. Hand plot each pair of closed loop poles in the complex plane (on the same plot). Indicate the number that corresponds to each gain next to the poles.

Table I.1: First Set of Gains

| Number | $K$ | Closed Loop Poles | P.O. | $t_p$ | $t_s$ |
|---|---|---|---|---|---|
| 1 | 0.002 | | | | |
| 2 | 0.01 | | | | |
| 3 | 0.1 | | | | |
| 4 | 10 | | | | |

3. Plot the root locus diagram for this proportional feedback system as $K$ is varied from 0 to $\infty$ using the standard root locus rules. Describe how the system step response would change as the gain $K$ is increased from a very small value to a very

large value. Be as specific as you can. Make sample sketches of the step response for a very small gain and for a large gain.

4. You want to select $K$ so that the system step response has the smallest settling time, while also maintaining less than a 5% overshoot. Where would be the best closed loop pole locations? Explain your answer carefully.

**Checking Results with Matlab**

5. Open the **CL_Constants.m** file from the *Closed Loop Step Response* experiment.

6. Save the file as **RL_Constants.m.**

7. Press the **Run** button  at the top of the page. Navigate to the **MATLAB** command window. Under “Workspace" on the right-hand side of the page, all of the variables from **RL_Constants.m** should be listed.

8. In the command window, type *g=tf([Km/tau],[1 1/tau 0])*. This defines the motor transfer function.

9. Now you will use a MATLAB tool to simplify the design process. (See `http://www.mathworks.com/help/control/getstart/siso-design-tool.html` for a detailed description of this tool.) Type *controlSystemDesigner* in the MATLAB command window. (Depending on the version of MATLAB that you have, you may need to use the command *sisotool* instead.) You should see the windows shown in Figure I.3 (depending on the version of MATLAB that you are using).

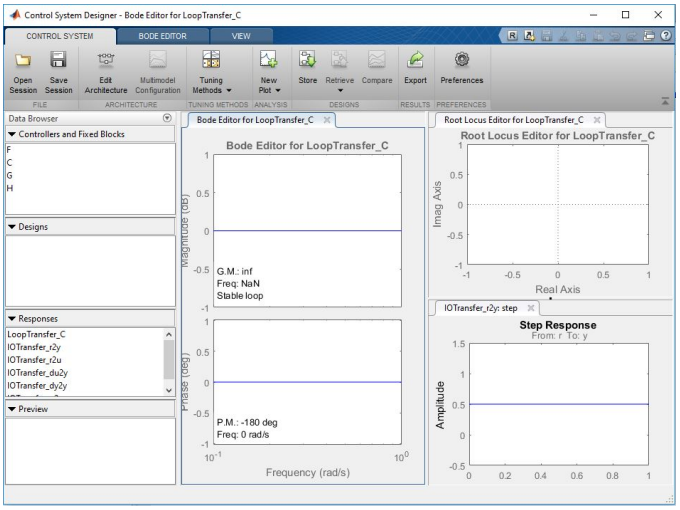



Figure I.3: Control System Designer Window

10. Click the **x** on the *Bode Editor for LoopTransfer_C* window to get rid of the plot.

11. Drag the rest of the plots to the left to make them bigger.

12. Click the **Edit Architecture** button to add transfer functions to a block diagram. Once the window pops up it should give you a figure that looks like Figure I.4. It shows the standard feedback control block diagram. For this experiment, there will be no pre-filter, so the $F$ block will be left as 1 or $< 1x1zpk >$. The $G$ block is the motor transfer function, and the $H$ block represents the measurements, which will also be 1 for the proportional feedback system we are considering in this exercise. The $C$ block represents the compensator, which will be the gain $K$ for our proportional feedback system.

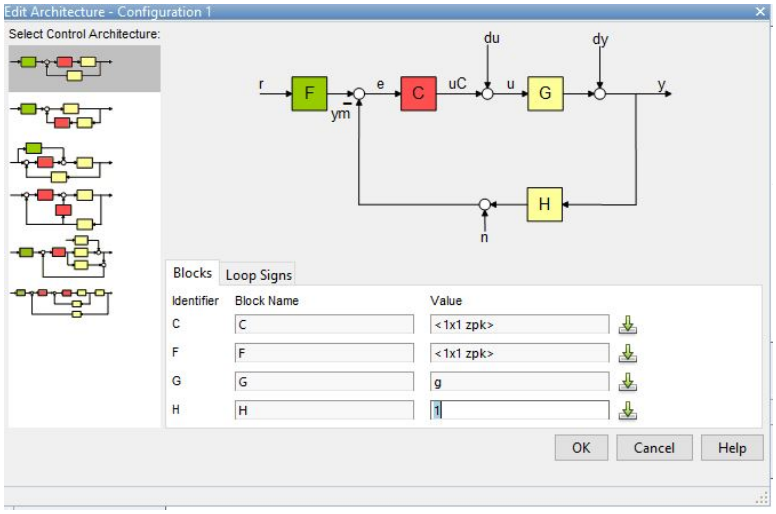



Figure I.4: Control and Estimation Tools Manager

13. The next step is to enter the motor transfer function into the *G* block of the *Edit Architecture Tool Manager*. Double-click in the *Value* column of the *G* row, and enter *g*, as shown in Figure I.4. Also, click in the *Value* column of the *H* row, and enter *1*. Then click *OK*. The root locus diagram should now be visible in one of the windows.

14. The step response that is shown will be for the default gain value of $K = 1$, since we did not change the default compensator value in the System Data window. The pole locations for this gain will be shown as small squares on the root locus plot, as shown in Figure I.5. (Your root locus plot may look different than this figure, since you have a different motor transfer function.) You can grab the small square and move the closed loop poles. This will cause the gain $K$ to change. (If you click on the *C* in the *Controllers and Fixed Blocks* subwindow at the upper left of the *Control System Designer*, the gain value will be displayed in the lower left *Preview* subwindow.) At the same time, the step response will change in the step response window. Save the root locus diagram for your lab notebook, and save the step response plot for a few different gain values. Discuss how these plots relate to the

root locus and step response plots you made in Step 3.

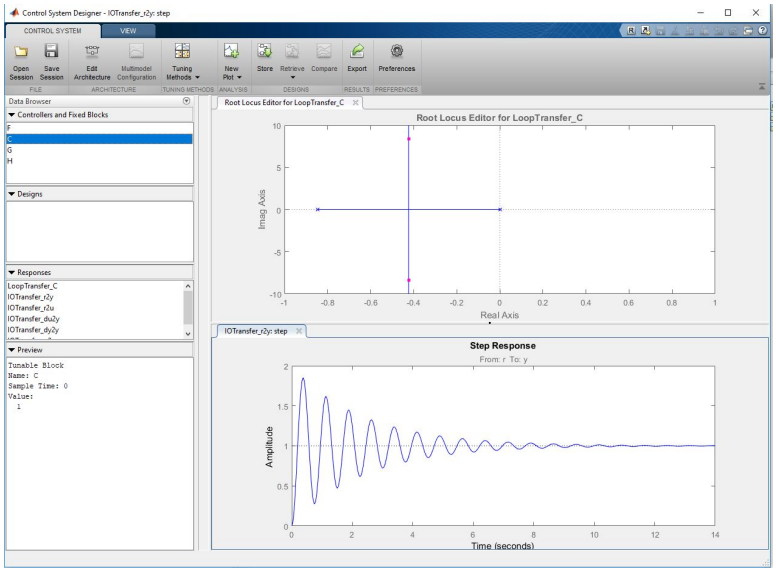



Figure I.5: Root Locus

15. By moving the closed loop poles, and monitoring the step response, select the value of $K$ that you believe will produce the best response in terms of smallest settling time, with minimal oscillation. Justify your choice. Save the best step response plot for your lab notebook.

16. For the $K$ that you selected, determine the voltage that it would produce, if the error $(r - y)$ is $\pi/2$. (Remember that $u(t) = K(r(t) - y(t))$.) Is this enough voltage to move the motor? Think back to the *Simple DC Motor, Open Loop Step Response* and *Closed Loop Step Response* experiments. Keep this in mind, when you analyze the experimental results later in this experiment.

### I.3.2 Exercise 2: Simulated Step Response (Proportional Feedback)

You will now simulate the closed loop step response before finding the step response experimentally.

**Setting Up Simulink File (Simulation)**

17. Open the **CL_Simulation.slx** file created in the *Closed Loop Step Response* experiment. It should appear as in Figure I.6. Set the simulation time to 10 seconds.

18. Save the file as **RL_Simulation.slx.**

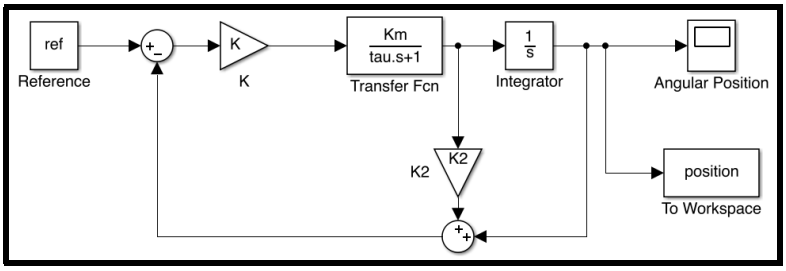



Figure I.6: Final Simulink Simulation Model

19. Open **RL_Constants.m** and set the value of $K$ to the value you found in Step 15. Be sure the value of $K2$ is set to zero. Then press the **Run** button at the top of the page. Navigate to the **MATLAB** command window. Under “Workspace" on the right-hand side of the page, all of the variables from **RL_Constants.m** should be listed.

20. Open **RL_Simulation.slx.** Click the Run button at the top of the page.

21. Once the model has finished running, double-click on the **Angular Position** scope block. Click the **Autoscale** button . Observe the plot. Does the closed loop step response appear to rise up from zero and settle to the reference value (as in your plot from Step 15)? If the plot looks to be correct (with a run time of 10 seconds) continue to the next step. Otherwise, go back to the previous section to ensure your Simulink file is correct.

22. Navigate back to the **MATLAB** command window. Under “Workspace" a variable (**position**) should now be available. Right click on **position** and click “Save

As..." Navigate to the folder in which you have saved this project, type next to “File name:" **RL_position_1.mat**, and click “Save" at the bottom of the page.

23. You now have the simulation data found from Simulink for the best proportional feedback controller.

**Simulation Plot File**

24. Open the main Matlab 2017a window and click **New** at the top and then click **Script**.

25. Once the new Untitled m-file appears, Click **Save**  at the top of the page. Save the file as **RL_Plot.m**.

26. Copy and paste the text in Listing I.1 into the Matlab file. After adding the code click **Save**  and then click **Run**  .

27. Save the figure as **RL_S_1.fig** into your folder for this project. Refer to this figure for the remaining steps in this section.

28. Compare the simulation results with the plot you found in Step 15. They should be almost identical. If not, then you will need to check the gain value you found in Step 15.

Listing I.1: Code for Plotting the Closed Loop Step Response Simulated Results

```
%Load the Simulation data and time and store into variables
RL_simResp_1 = load('RL_position_1.mat');
t = RL_simResp_1.position.Time;
RL_simResp_1 = RL_simResp_1.position.Data;

%Plot the simulation data with respect to time
figure;
plot(t,ones(size(t))*ref, 'Color', 'r');
hold on;
plot(t,RL_simResp_1, 'Color', 'k','LineWidth', 2);
```

```
title('Simulated Closed Loop Step Response');
legend('Reference','Simulated', 'Location', 'southeast');
xlabel('Time (seconds)')
ylabel('Theta (radians)')
```

### I.3.3 Exercise 3: Experimental Step Response (Proportional Feedback)

This section will provide the setup of the Simulink file for the Arduino.

#### Setting Up Simulink File (Arduino)

29. Open the Simulink file created in the *Closed Loop Step Response* experiment named **CL_Step_Resp_Arduino.slx**. It should look like Figure I.7

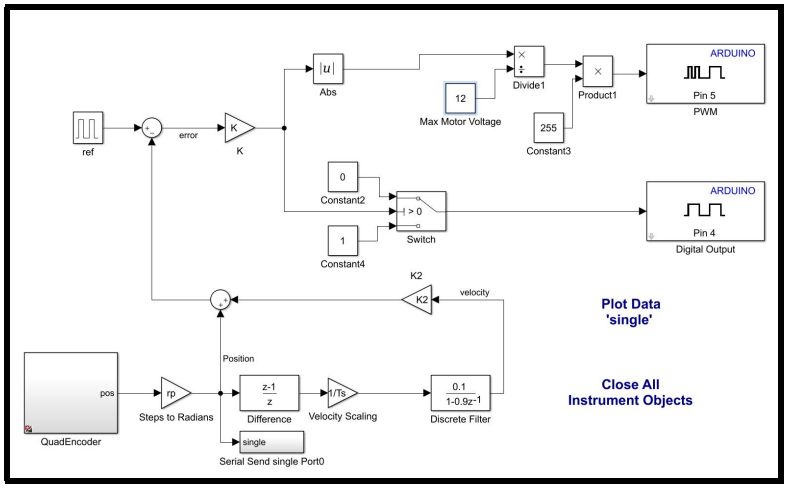



Figure I.7: Closed Loop Simulink Model for Closed Loop Step Response Experiment

30. Delete the **Discrete Filter** block and connect the line from the previous **Velocity Scaling** to the **K2** block.

31. Click **File → Save As... → RL_Step_Resp_Arduino.xls**. The Arduino Simulink file for the experimental closed loop step response (proportional feedback) is now complete. See Figure I.8 for the completed model.

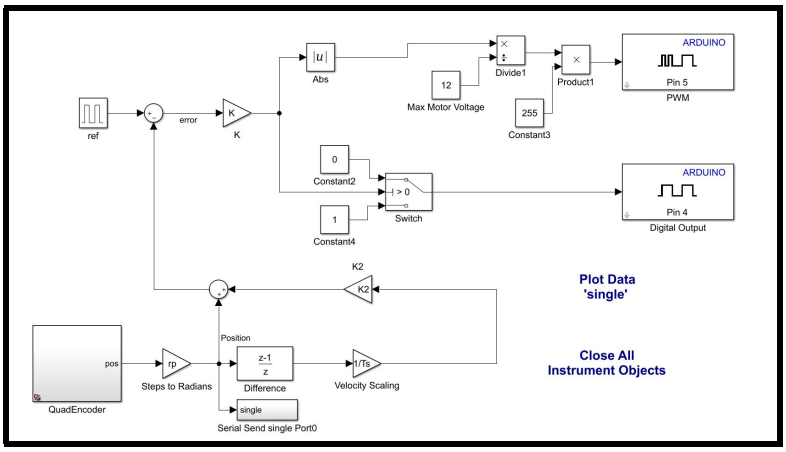



Figure I.8: Final Closed Loop Step Response Simulink Model for Arduino

**Collecting Experimental Data**

32. Open **RL_Constants.m** and click the Run button  at the top of the page.

33. Open **RL_Step_Resp_Arduino.slx** and click the “Deploy to Hardware" button  at the top-right of the page.

34. Once the model has successfully deployed to the Arduino, double click on the text **Plot Data ‘single’** inside the model window.

35. When the small window labeled “Plot Ser..." appears, enter the Arduino COM port number under “Enter COM port to collect data:." The default values for “Enter Number of Samples to plot:" is “single" and for “Enter Number of samples to plot:" is **8000**.

    **Note:** To find the COM port number for your Arduino, refer to the *Simple DC Motor* experiment under the section “Software Setup → Installing Arduino Mega 2560 Drivers."

36. Click Okay. Once the plot appears, plug the power cord from the power supply into the motor shield.

**CAUTION:** Do not put your hands or any other parts of your body in front of the motor load trajectory.

37. The motor should attempt to turn 90 degrees and stop. (The motor may not actually turn a full 90 degrees, depending on the gain value that you used for the controller.) Then it should return to its original position and stop. This cycle should repeat itself every 20 seconds. If the plot does not reflect the movement that you see in the motor load, follow the steps you used in the *Closed Loop Step Response* experiment to obtain a reasonable plot.

38. Let the data fill the plot window as it moves to the left. Once the data has filled the screen completely *and the first pulse has moved to the left off the screen*, click the “Stop" button at the bottom of the screen. You should now have 8000 data points on the screen. **Note:** you should at least let the first pulse disappear as it usually will not be the full ten seconds.

39. Navigate back to the **MATLAB 2017a** main page. Under “Workspace" the variable **WindowDat** should now be present. Right-click on it and click “Save As." Name the file **RL_expResp_1.mat** and save it into the folder where the **RL_Step_Resp_Arduino.slx** file is saved.

40. You now have the experimental data for the closed loop step response with proportional feedback.

**Experimental Plotting File**

41. Open the **RL_Plot.m** file you created in the **Simulation Plot File** section.

42. Add the text in Listing I.2 to the bottom of the **RL_Plot.m** file. After adding the code, click **Save**  and then click **Run**  .

Listing I.2: Code for Plotting the Closed Loop Step Response Experimental Results

```
%Load the experimental data and store into a variable
RL_expResp_1 = load('RL_expResp_1.mat');
RL_expResp_1 = RL_expResp_1.WindowDat;

%Align the experimental data with the reference
%and compute the root mean square error.
[yplot,minrmse,~,~] = findShift2(RL_expResp_1,T*(1/Ts),D*T,ref);
T1 = Ts*(0:(length(yplot)-1));

%Plot the experimental data
hold on;
plot(T1,yplot,'b','LineWidth', 2)
ax = axis;
text(ax(2),ax(4)-0.1,['Experimental RMSE = ' num2str(minrmse)],...
'HorizontalAlignment','right','VerticalAlignment','top');
legend('Reference','Simulated','Experimental','Location', 'southeast');
```

43. Save the figure as **RL_SE_1.fig** into your folder for this project. Refer to this figure for the remaining steps in this section.

44. Compare the simulation results with the experimental results you found. Estimate the settling time, percent overshoot, and frequency of oscillation of the closed loop step response from the experimental plot. Compare with the simulated plot. Also compare the steady state values from each plot, discussing similarities and explaining differences. What could cause the experimental response to differ from the simulated response? Are there nonlinear effects in the motor that could change the performance?

45. Can you adjust the gain $K$ to improve the experimental response? The root mean square error (RMSE) between the reference and the motor angle is shown on the experimental plot. How small can you make this value by changing the gain $K$. Go back to the earlier steps when you selected the $K$ value. Try different closed loop

pole locations. Perhaps you need to accept a larger percent overshoot in order to achieve a smaller RMSE. Explain your design process.

### I.3.4 Exercise 4: Control Design (Proportional plus Derivative Feedback)

In this exercise you will design a proportional plus derivative (PD) feedback controller for the DC motor, using the root locus diagram. The controller signal $u(t)$ (motor voltage) will be $K$ times $r-(k_2\omega+k_1\theta)$. Since $k_1$ will be set equal to 1, $K$ effectively multiplies $(r-\theta)-k_2\dot{\theta}=e-k_2\dot{\theta}$. The term $Ke$ is called proportional feedback, since it produces an input that is proportional to the error. The term $Kk_2\dot{\theta}$ is the derivative feedback, and has a damping effect, like viscous friction. The block diagram of the PD controller is shown in Figure I.9.

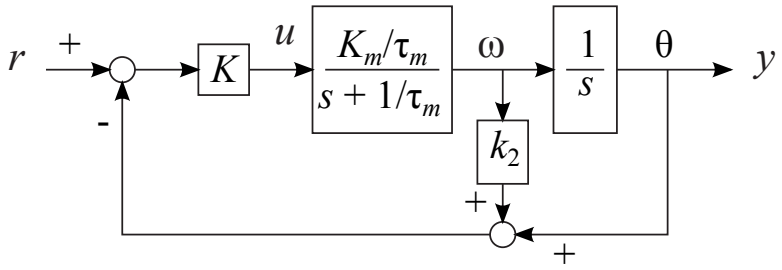



Figure I.9: Block Diagram for Closed Loop Motor with Proportional plus Derivative Feedback

46. Using block diagram manipulation on the block diagram in Figure I.9, find the transfer functions $G(s)$ and $H(s)$ for the equivalent block diagram in Figure I.2. Plug in the values for $K_m$ and $\tau_m$ that you found in the *Open Loop Step Response* experiment. Your $H$ transfer function should be in the form $k_2(s+b)$

47. Let $k_2=0.2$, find the closed loop transfer function, and find the closed loop poles as a function of $K$. Complete Table I.2 and hand plot the closed loop poles for each gain (on the same plot) denoting the number that corresponds to each gain next to the poles.

Table I.2: Second Set of Gains

| Number | K | Closed Loop Poles | P.O. | $t_p$ | $t_s$ |
|---|---|---|---|---|---|
| 1 | 0.002 | | | | |
| 2 | 0.01 | | | | |
| 3 | 1 | | | | |
| 4 | 2 | | | | |

48. Let $k_2 = 0.2$, and plot the root locus diagram for this proportional plus derivative feedback system as $K$ is varied from 0 to $\infty$. Describe how the system step response would change as the gain $K$ is increased from a very small value to a very large value. Be as specific as you can. Make sample sketches of the step response for a very small gain and for a large gain.

49. You want to select $K$ so that the system step response has the smallest settling time, while also maintaining less than a 5% overshoot. Where would be the best closed loop pole locations? Explain your answer carefully.

50. If you change the value of $k_2$, how is the root locus affected? Use sketches of the root locus for various values of $k_2$ to illustrate the effect. By adjusting both $K$ and $k_2$, how much flexibility do you have in placing the closed loop poles? Are there theoretical limits on the closed loop pole locations? Are there practical limits on the closed loop pole locations? Discuss these ideas in detail.

**Checking Results with Matlab, Simulation and Experimental Results for PD Control**

51. Repeat Steps 7 to 45 for the proportional plus derivative feedback system. You will need to create the $H$ transfer function (like you created the $G$ transfer function in Step 8) and load it into the *Control and Estimation Tools Manager* (like you did

for the $G$ transfer function in Step 13). You will also need to modify the values for $K$ and $K2$ in the **RL_Constants.m** file. (When you are saving figures and data files, you will want to adjust the file names, and use these new file names in the **RL_Plot.m** file.)

52. After completing the simulations and experimental results for the PD controller, experiment with different values for $K$ and $K2$. Can you reduce the steady state error, while maintaining a low overshoot and minimum settling time? Find the controller that produces the minimum RMSE. How much lower can you make the RMSE using the PD controller, when compared to the proportional controller? Explain your final tuning process and justify your final design. Discuss theoretical aspects of pole locations and their relation to overshoot and settling time, and explain practical considerations that must be taken into account to reduce steady state error when nonlinear effects must be taken into account.

53. What is the resolution of the encoder in radians, if there are 64 counts per revolution? How big is the steady state error for your experimental results? Can you make a connection between the encoder resolution, which is used to measure the motor angle, and your steady state error?

### I.4 Table of Discussions and Questions

Before you turn in your report for this experiment, make sure that you have answered all of the questions that have been posed. It is important that your answers be expansive and that they demonstrate that you were mentally engaged in the experiment. Below is a recap of the important questions and the number of the step where each question was embedded.

| steps | Discussion/Question |
|---|---|

| | |
|---|---|
| 1 | $G(s)$ and $H(s)$ transfer functions |
| 2 | Table I.1 and hand plot of closed loop poles |
| 3 | Plot root locus |
| 3 | System response as K is increased |
| 3 | Sketches of step response for a very small gain and large gain |
| 4 | Best pole locations and selection of K |
| 14 | Step response plots and root locus for different gains |
| 14 | Comparison with Step 3. |
| 15 | Selection of K and step response plot |
| 16 | Voltage if error is $\pi/2$. Is it enough voltage? |
| 21 | Does your response appear to rise up from zero and settle to the reference value? |
| 44 | Comparison between simulation and experimental results |
| 44 | Settling time, percent overshoot, frequency of oscillation and compare with simulation |
| 44 | Similarities and differences |
| 44 | What could cause the experimental response to differ from simulation? |
| 44 | Are there any nonlinear effects in the motor? |
| 45 | Explanation of design process for making RMSE smaller |
| 46 | $G(s)$ and $H(s)$ transfer functions |
| 47 | $k_2 = 0.2$ Table I.2 and hand plot of closed loop poles |
| 48 | Plot root locus |
| 48 | System response as K is increased |
| 48 | Sketches of step response for a very small gain and large gain |
| 49 | Best pole locations and selection of K |
| 50 | Changing $k_2$. |
| 50 | Flexibility of closed loop poles |

| | |
|---|---|
| 50 | Theoretical and practical limits |
| 51 | Make sure you answer all the of the questions (should be similar to previous in the table) |
| 52 | Discussion on process of finding gains and final gain parameter design choice and calculations. |
| 53 | Encoder resolution |
| 53 | Connections between steady state error and encoder resolution |

### I.5 Conclusion/Student Feedback

This experiment lead you through the design process for proportional and proportional plus derivative feedback controllers. The PD controller enabled more control over the placement of closed loop poles, and allowed an improved system response.

## APPENDIX J

### Free Response Questions

Table J.1: Simple DC Motor Free Response

| Response |
| --- |
| The part that took the longest was tracking down and downloading all of the required software. Everything else was pretty straightforward and went step by step with the instructions. It was a little cumbersome trying to find the website with the lab information on it, but it is bookmarked now, so that shouldn't be an issue any longer. |
| I really liked seeing the motor react to my commands on Matlab. I would say the most cumbersome parts of the lab were the parts that started talking about terms I am not familiar with (as indicated in my submitted lab report). Overall, I thought this was a nice introductory lab. |
| Aside from attempting to get my schematic to run, the part that took the longest was step 69, where we had to change the voltage input to the motor. It wasn't too long, however, as it was relatively simple to do. It was interesting to see that the motor had to overcome static friction. It was also amusing to see the motor move rapidly with the 8 volt input. |
| The difficult part of this project is to get familiar with the new SimuLink toolbox and understand how the system works. But I think after this warm-up project, we will get comfortable with them. |

| |
|---|
| The part of the lab that took the most time for us was setting up matlab, which I suppose was an error on our end for using an outdated version of matlab. The most interesting part for me was using simulink, and seeing how simple it was to make our basic simulink schematic come to life in front of when we were given guidance on how to do so. |
| The most cumbersome part of the lab was the setup of the Simulink modules. The most interesting part of the lab was finding the voltage at which the motor overcame the static friction. |
| The most time consuming part of the lab was downloading and properly configuring the software. The most interesting part of the lab was how easily controlled the motor was. |
| the part of the lab that took the longest was trying to set up all the simulink. I thought matlab2016 would do the job but it did not so I had to reinstall matlab2017. the most interesting part of the lab was spinning the fan and seeing how was it can spin a different voltages. |
| There were no really lengthy parts of the lab.It all seemed smooth from one step to another with none taking too extremely long compared to the others. In my opinion, having 8 pennies added into the load for the motor and applying 8V created a potentially dangerous system. I took precautions when spinning the motor that fast by putting a shield around the blade and thankfully so because one set of pennies flew off into the shield. The most interesting part of the lab was probably finding by trial and error the lowest voltage the motor would spin. |

| |
|---|
| Lab was mostly straightforward, albeit, my original understanding of the briefing instructions in lecture indicated that only Section 3 was pertinent to the completion of the lab (since kit was preassembled). This was not the case, as Section 2 had specific instructions regarding the configuration of MATLAB/Simulink. Wasn't expecting the noise, but that is a nonfactor. It did take longer than expected for MATLAB/Simulink to build and deploy. |
| Question: What took the longest and was most cumbersome? Answer: Figuring out how to use the simulator software to represent the motor took the longest and was most cumbersome as I had never used that Simulink package before. Question: What was the most interesting? Answer: What I found most interesting was stepping the voltage to find the amount required to over come the frictional resistance of the motor for the fan to begin to rotate. I did not know before this experiment that there could be a different frictional resistance threshold for the different directions of the motor. |
| The lab was well written and provided the steps clearly. I got through the lab pretty quickly. Writing the lab report and answering the questions was what took the longest for me. I found controlling the motor through Simulink very interesting. It was cool to see the motor speed up and change directions when I changed the numbers on simulink. |
| The longest part of the lab was definitely the software setup. The downloads took a lot longer than I thought they would. The most interesting part of the lab definitely began when we started to spin the motor. I enjoyed the portion of the lab when we changed the direction. I had no idea that a 0 or 1 could change the direction the motor would spin. |

| |
|---|
| The longest part of the lab for me was getting all the software downloaded and the set up done. After that the actual execution of the lab took about 30 minutes. The most rewarding part was watching the motor start to turn as the voltage was slowly increased. |
| The most cumbersome part of the lab was downloading the software, and that was not bad either. The most interesting part of the lab was setting up the Simulink control and how simple it is. It seems more intuitive than LabVIEW. |
| The lab was good, however, the questions and the lab, in general, could have been worded better in order to clearly ask what was needed. There was a high health risk when the motor spun too fast and pennies started flying out of it. Another thing would be a clear outline of what the report needs to look like in future labs. |
| Following through the instructions and downloading the software was the most cumbersome portion of lab1. |
| The lab was pretty simple but the set up took the longest. The parts that took the longest were getting the schematics drawn correctly. During my first run I did not do this correctly and could not figure out what was wrong. In the future I will make sure I pay attention to every single step so I will not run into this problem. |
| The most time consuming part of the lab was making sure everything was set up correctly. I had Matlab R2015b and was not able to download the arduino simulink package no matter how many times I attempted to. I didn't get it to work until I downloaded R2017. The detailed instructions with corresponding screenshots were very helpful for getting the software ready. The steps were straight forward and easy to move through. Overall, a good way to begin understanding how the motor functions with PWM. |

| |
|---|
| The most time consuming part of this lab was probably the set up of the software. Otherwise it was pretty straight forward. The only portion of the lab that was somewhat confusing was solving for friction. It was unclear what we need to observe to know when we found the correct value. Overall, I really liked the lab. It applied a few concepts from class and it was cool seeing how all of the software and hardware worked together. |
| Finding the threshold voltage in both directions was the longest part to me. Learning to work with the board and motor was very interesting. |
| The part that took the longest time was testing for the lowest voltage value to overcome the static friction in both directions. The most interesting part was seeing how Simulink turned a model into code for the Arduino and how easy it was to adjust and change. |
| The longest part of the lab was just downloading the software. It took significantly longer than I was expecting. It was good to have a chance to get comfortable with the software though. |
| The most time consuming part of the lab was waiting for the code to upload to the Arduino. Most of the time spent was on this. The most interesting part of the lab was seeing how the pwm could produce the same output as a step function. |
| Setting up the software had a few hiccups due to the fact that Matlab version 2016a on my laptop would not run the RC script. But after a script upgrade from the TA it worked. The interesting part of the lab was able to change the speed of the DC Motor using simulink and the arduino. |
| The lab in general was relatively smooth to complete. However, manipulating Matlab to SimuLink and switching between softwares was frustrating to keep track of. |

| |
|---|
| The lab went very smoothly, and was very easy to follow, however, to find the voltage point to overcome motor friction, we had to build the code more than 2 dozen times, each build taking at least 1 minute. I am not familiar enough with simulink to have programmed a way to change the voltage on the fly, but that is what I should have learned and done. Interacting with and using simulink like labview was very interesting. I am glad to have the opportunity to use such advanced software and refined take home labs. |
| The part of the project that took the longest was downloading all the needed software. The most interesting is the effectiveness of Simulink to transition a created diagram to usable code for the arduino. |
| The first lab is super easy. we spent some time to set up the digital blocks, and I think setting up the blocks and fallowing the instructions is the longest part. The most interesting part is when the load or wheel start moving, and how much the wheel's speed become when we increase the voltage. |
| This was a very good lab. I do not think much of it was cumbersome. The only part that seemed cumbersome was taking the time to set everything up but of course it needs to be done. Dealing with errors is probably the biggest problem I had. I found it interesting to see how a slower modulation and faster modulations alter the response. |

Table J.2: Sampling and Data Acquisition Free Response

| Response |
| --- |
| The hardest part of this lab was getting the library to work on Mac. Trevor was able to help me get this to work. The best part of this lab was at the end seeing the experiment and our calculations match. |
| The parts most cumbersome were dealing with the errors and correct setup for the lab. The experience overall was beneficial. It inspired me to ask questions of my own. |
| The longest part was fixing the software for macOS. |
| I think the most time consuming part of the lab was getting the code and software to work properly. The most enjoyable part of the lab is messing around with the motor and watching it do what it is supposed to do. |
| This was a good lab, not much to complain about it. |
| The longest part of the lab was probably researching signal sampling to understand the concept, as I have not yet covered this subject in any of my classes. It was interesting to see how different sampling rates affected the output, especially those that were too low to accurately represent the original signal. |
| I think the best/most informative parts of the lab was getting the all the different graphs input points from the motor. Combining matlab and a dc motor and getting actual useful information instead of just odd simulations like in other classes. The part that could be better is that some of the instructions for building the matlab model were not very specific. Finding the quad encoder for instance was not very clear. |

| |
|---|
| I enjoyed this lab. I felt like it flowed really well. I liked that the topics built on each other as you went along. We started with the hand sketches which made sure we knew what the original sine wave looked like and also what the different sampling rates should look like. After that, we used simulink to get the same waveforms which reinforced the idea further. And the example where we set the sampling rate to 0.9 and the output looked like one sine wave when it was actually multiple sine waves was really interesting to me. I had some trouble loading the code into the arduino for awhile but that was the only thing that took too much of a long time. |
| the only part of the lab that took long was trying to get the THL library to work with my computer. i guess my computer kept placing in another spot. |
| I didn't encounter any part that I felt was overly cumbersome. It was very interesting getting a better understanding of using Matlab and Simulink and applying them. |
| Getting through the Mac-specific bugs took the longest. Acquiring data from the motor was most interesting. |
| The part of the experiment that took the longest time and was mots troublesome was that there were issues with accessing the Take Home Labs Arduino Support Package library. We had to add it multiple times to get the added parts to work correctly in the simulated designs. I found it most interesting that when we changed the step size to .9 and the end time to 10 seconds we received a complete different signal from the original signal. I had not realized before this lab that you could destroy your original signal by taking an improper sampling rate. |
| Taking the data for the experimental sampling rates and using the data to produce the velocity figure took the longest time. The most interesting part was how the serial plot initially took data from the motor despite not being plugged in. |

| |
|---|
| The lab was very straightforward and easy to understand. However, I was confused at the end by the serial feed looking the same as the graph that was created once the feed was stopped and the DCmotorDATA file was run. I was unsure as to whether they were supposed to look the same, and the graph generated with time on the x-axis was so that the data could be viewed in time. Should the graphs have looked different? Otherwise, I had no problem conducting the experiment and received no errors. The detailed instructions are very helpful. |
| The most cumbersome part of the lab was the setup of the software. The most interesting part of the lab was watching the sampling take place while the motor was rotating. |
| Through this project I had a better understand about how the sampling frequency affect the result. The lab is easy to implement and interesting for learning. |
| Overall, lab was easy to follow. Unclear why data sampling resulted in different results from simulation and hand sketches. Building the model was relatively straightforward, albeit ensuring the sampling period was correctly set was more crucial than expected for proper operation. |
| The part that took the longest was the part about syncing the arduino and the computer because I forgot to follow the steps from the previous lab. The most interesting part of the lab was actually getting the graphs from the arduino and computer relation. |

Table J.3: Open Loop Step Response Free Response

| Response |
|---|
| Solving for the transfer functions took the longest. Once I understood how to solve for G1, it was "smooth sailing." The most interesting part was seeing how the experimental response was affected by the inertia of the load. I think it would be a great idea to suggest changing the inertia just so students can see that it does affect the response. |
| The most difficult aspect to this lab was deriving the equations and transfer function. Once I decided to put all the equations in terms of w, everything fell into place. |
| The hardest part of the lab was the beginning. This was due to the fact that we did not cover the material until the tuesday before the lab was due. |
| I spent most time in the error of findshift2.m. I changed 6000 to 12000 but still got the error. Then I tried re-sampling for 6 times and succeeded by accident finally. But I still don't know what happened with the error in findshift2 function. |
| The most time consuming part of the lab was getting the software to run correctly. The most interesting part of the lab was getting theory and experiment to match up on. |
| I thought the most interesting part of this lab was the process of identifying the proper time constant. I struggled towards the beginning on finding the transfer function, but I simply over-complicated the problem. Otherwise, overall, it was generally a interesting lab. |
| The most challenging part of the lab was calculating the values for the transfer function and proving the equivalency of the equations from the transfer function. |
| The most cumbersome part of the lab was all of the calculations. The most interesting part was seeing how the simulation and the actual experimental plot resembled each other. |
| Following the steps to get simulation on the MATLAB took the longest time. Writing the report and get summary of the whole lab is more interesting for me |

| |
|---|
| Generally, the whole experiment was straightforward and only deriving the equations and solving by hand took the longest time. However, setting the comparison graph was the most interesting part as it shows how both results were almost matching for the response of our design. |
| The most time consuming part of the lab was setting up the experimental and simulation stuff. The most interesting part of the lab was solving the math. |
| The hardest/ most cumbersome part of this lab was all of the theoretical calculations. Once I got a handle on that, the lab went smoothly. The most interesting part of the lab was comparing the experimental and simulated plots. I was excited to see that my two plots were almost exactly the same. It was a cool way to check that my calculations were correct. |
| the part of the lab that was most time consuming was trying to figure out the transfer functions. |
| Longest part was deriving the transfer functions and trying to figure out the fastest way to solve for the equations to get what you need to make the time function in MATLAB |
| The beginning of the lab took the longest, it was a little cumbersome to acquire the transfer functions for some of the blocks. The most interesting part of the lab was getting to see how the things we've learned in class directly apply to a physical application. Getting to solve for the pole of the system and observe how it was affected by the added weights or fins. |
| Setting up the Simulink for the simulation and setting up matlab for the experimental results were the most time consuming part of the experiment. Collecting the experimental results was the most interesting part. Seeing how slight vibrations of the motor and mount will change the response graph results. |
| The hardest part of the assignment was deriving the formulas. The most interesting part was how well the simulation matched the experimental results. |

| |
|---|
| Preparing the report itself took the longest because a lot of equations were involved. I taught myself how to type equations with Latex just to save a bit of time. It was interesting to me how close my calculations were to the experimental results. |
| There did not seem to be one part of this lab that took longer than the rest. The main issue I had was that a few of the conceptual questions were hard to understand. |
| The hardest part of the lab was deriving the transfer function and relating it to the other variables. The most interesting part was actually getting the transfer function from the hardware. |
| Because our motor was loose, we had to hold it down while we were recording our plots which was really cumbersome. The most interesting part was how our simulation and experiments were very close to each other. |
| the longest part was finding the transfer function from the blocks, because we had started the lab before we learned the reduction block. the most interesting part was Comparison of experimental and simulated response |
| The longest and most cumbersome part was doing all the math and rewriting all the equations. The most interesting was comparing the experimental data against the simulated data. |
| The lab was a little bit easier than the previous lab, however, it would have been nice to have the load assignments earlier so I wouldn't have to redo the lab if I got the load wrong. Otherwise, it was a good lab. |
| Calculation of what need to. I am not good at calculating |

| |
|---|
| The hardest part of this lab was when multiple different parts of the code didn't work, and we tried debugging several different times before having to redownload some files. Our Simulink plot wouldn't take all the data points at a time so we thought we had done the experiment, but we had to redo it when we realized it wasn't working correctly. The most interesting part was seeing the how the different amount of pennies affected each person's outputs differently and in ways that I didn't expect. |
| I think the hardest part of this lab was correctly solving for the transfer function using the block diagram and the equations for the mechanical system. |

Table J.4: Closed Loop Step Response Free Response

| Response |
| --- |
| The most cumbersome part of the lab was the calculations at the beginning, and the most interesting part was seeing how the experimental data compared to the simulations. |
| The experiment didn't take me long time. All the software parts have been set up previously. The most confusing thing is that the experimental plot does not consist with the simulation. I recalculate the equation many times to make sure every coefficients are correct but still got the problem. |
| The lab felt surprisingly short compared to the others which may be because it was similar to the previous lab. The most interesting part of the lab was definitely when it had us attempt to control the motor "manually" using the potentiometer. |
| the part that took the longest was having to redo the simulation and experimental for the second case. the most interesting part was trying to turn the fan 90 degrees. |
| Step 55 of the lab, manually turning the motor exactly 90 degrees and stopping, took the longest time. However, discovering the similarities and differences from theory, experiment and simulation was the most interesting part of the CLSR lab. |
| The most interesting part of the lab was comparing the experimental and simulated results. It was interesting to see the differences between these. The hardest part was the calculations at the beginning, I struggled with finding some of the formulas for the values we were supposed to find. |
| The part of the experiment that was the most cumbersome was the setup and deployment in order for the Arduino to operate the motor. The most interesting part of the experiment was how big the difference between the overshoot was between both experiments ran and also how difficult it was to manually stop the motor at only a 90 degree turn. |

| |
|---|
| The most time consuming part of the lab was getting the software to cooperate, the most rewarding part was seeing the theory and experiment match up |
| The hardest part of the lab was making sure i had the correct K value. After I redid my Open loop Lab and found a new K value i was able to understand the closed loop lab The hardest part of the experiment was simplifying the block diagram. |
| The most challenging part of the lab was the doing the calculations/theory at the beginning. The most interesting part of the lab was seeing that the experimental results and simulations didn't match, and trying to figure out why. |
| I found the theory part to be the longest most cumbersome part of this experiment. However I found it to be very interesting when we were dealing with the different plots and I enjoyed plotting the 2nd order time response by hand. |
| Drawing the graphs probably took the longest, but this might be because I didn't understand them at first. I think the most interesting part of the lab was seeing the motor with and without K2. It was cool to see the motor pretty much just stop at 90degrees when K2=0.1. |
| The parts of the lab that took the longest was the theory portion at the very beginning. Manipulating the block diagram at the beginning was quite cumbersome due to the size of the fractions that were in the blocks.The most interesting part of the lab was seeing the motor rotate exactly to 90 degrees, even with user input affecting the position. |
| The theory (solving everything out by hand) was the most cumbersome part of this lab part. |
| The easiest part of the lab was the part where we had to try and turn it 90 degrees. The most difficult part of the lab were just minor details that I would miss with interfacing the arduino and computer. |

| |
|---|
| The parts of this lab that took the longest were the block diagram simplification and using Matlab effectively. The block diagram took a while because we made an early error. I feel as if I should have figured out matlab by now, and to an extent I have, but my Matlab has become extremely messy because of my tendency to click "add to path" rather than "change folder." This was my first real Matlab experience, but after doing these labs I feel I have a much better handle on it. The part I found most interesting was how difficult it was to make the motor stop after exactly 90 degrees. This was much more challenging than I suspected it would be, and gave me an appreciation for complex mechanical and electrical systems such as cars by showing me how little I know about basic concepts such as motors. |
| Analyzing the percent overshoot, settling time, frequency and steady value took me the longest time. The interesting part is to get motor turn and stop at eaxct 90 degree |

Table J.5: Root Locus Control Design Free Response

| Response |
| --- |
| The most difficult part of the lab was trying to find out which K value would give the best result via the root locus diagram. Theory did not always match the experimental results. The most interesting part of the lab was watching how changing the K value changed the closed loop pole locations and step response. |
| The most interesting part of the lab was the simulation part, including when we got to hand select the K value. The most cumbersome thing on the lab was the theoretical sections. |
| I struggled early on with the transfer functions because I got extremely confused on the directions with the block diagrams. It may appear clear to the instructors who are grading the lab, but when I initially went over the instructions I couldn't figure out what it was asking me to do. The figure that has F G H had me confused. This took me about two days to finally realize that H(s) could be just 1. At first this was misleading ,and I thought I was doing it wrong. The most interesting (also took extremely long) was the trial and error with differing K and K2 values. When I finally got the system to do what I wanted, it was very gratifying to know that I had successfully completed the lab. |
| The processing that I changed the value of K and k2 to get the smallest RMSE took me the longest time. The interesting part is to compare the hand sktech plot with the plot deigned in MATLAB. It gave me deep understading with the Root Locus |
| The root locus free response was a very interesting lab. I thought it was cool how you could use k and k2 to change the oscillations and output response. The hardest part was realizing that we couldn't use the "ideal" k values to run the motor because there was too much friction. It took me awhile to understand this. |

| |
|---|
| This lab was exceptionally hard because I was unable to contact my lab partner who had our team's hardware. This caused me to be delayed in starting the project as well as correcting sections of the lab notebook that were not done correctly. The lab itself was not extremely challenging, but doing it in the minimal amount of time I was allotted (I borrowed another group's hardware) was extremely challenging. I would also like to note that I left the fins with the hardware that I borrowed, so they will be turning it in. |
| The most time consuming part of the lab was all the calculations and playing with the k values to make the motor work better. The most interesting part of the lab was seeing how things we are tested over could possibly be used in practice. |
| The entire lab took some time. The theory, calculations, and hand made plots took a good amount of time. Also, entering the transfer function into matlab for H(s) was not explicit. I had to do the first half of the lab a few times to get acceptable data. I second part with velocity feedback was much easier after toiling so long with the first parts of the lab. |
| The most cumbersome part of this lab was having to upload different K-values to the arduino over and over, but I also found it to be the most interesting because it gave us a better idea of what K and K2 actually did |
| What took me the longest was getting simulink to recognize the Arduino port. I figured out that I had to restart my computer any time I wanted to deploy to hardware. I liked being able to see the root locus diagram and drag the poles to see how it affects the step response. |
| The most difficult parts of the lab for me was understanding the conceptual parts of the root locus system. To be honest, deriving the root locus is still something I am unsure about but this lab did help further the understanding. |

| |
|---|
| The theoretical portion for the derivative feedback system took the longest. The result of the derivative feedback was the most interesting of the lab because the hard work put into the theoretical portion felt like it was necessary and helpful. |
| The most difficult part of this lab was working out all of the theory. The simulation part did not take much time at all. This lab helped me fully understand the root locus and how to create a control system |
| The parts that took the longest time were the testing different Ks to find the best pole locations for the experimental results. The most interesting parts were seeing how the motor reacted to different Ks and K2s. |
| the longest part was doing the math part to fill out the tables.. the interesting part was finding the best value for K to run the motor in 90 degree |
| the most cumbersome of the lab is having to do the lab again for a second case. it take to much time. another issue was not knowing how the response need to look like. a example graph can help students know they are on the right path. the interesting part of the lab was watching the the time response will change as the the gain increased in the matlab program. |
| This lab was very interesting. Getting to manipulate the root locus and control the motor accordingly while getting to see the dramatic response changes was the most interesting part. The part that took the most time was having to repeat through the majority of the lab with the new set of feedback. That, and solving the theory at the beginning. |
| The most cumbersome part of the root locus lab was getting the second part of the simulation and experiment with matlab to work properly. The most interesting part was seeing how selecting a more accurate k value produced a more precise response from the motor. |

| |
|---|
| The most difficult part of the experiment was finding the best values for K and K2 both theoretically and experimentally. |
| There was really no long or cumbersome part to this lab. Once you got past the theory it moved pretty quickly. The most interesting part for me was getting a better understanding of how control systems are designed and understanding that sometimes specifications cannot be met depending on your hardware. |
| I think that this section took me a while to do just because I didn't fully understand what I was supposed to be looking for. I didn't fully understand the root locus until after I had already turned in the final project and was studying the 2nd exam in preparation for the final exam. |

VITA

Trevor Eckert

Candidate for the Degree of

Master of Science

Thesis: A DYNAMIC SYSTEMS TAKE HOME LABORATORY

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Tulsa, Oklahoma, United States of America on October 19, 1993.

Education:

Received the B.S. degree from Oklahoma State University, Stillwater, Oklahoma, United States of America, 2016, in Electrical Engineering
Completed the requirements for the degree of Master of Science with a major in Electrical Engineering at Oklahoma State University in May, 2018.